

L^AT_EX₂ ϵ

科技排版指南

邓建松 彭冉冉 陈长松 编著



科学出版社
Science Press

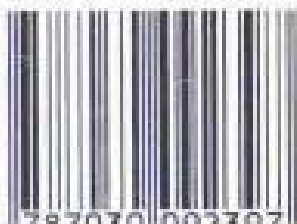
(TP-1516.0101)

责任编辑: 杨 波 封面设计: 王 浩

L^AT_EX2_ε

科
技
排
版
指
南

ISBN 7-03-009239-2



9 787030 092397 >

ISBN 7-03-009239-2/TP · 1516

定 价: 40.00 元

974

7.2.1.1
D.2

L^AT_EX 2_ε 科技排版指南

邓建松 彭冉冉 陈长松 编著



A0952450

科学出版社

2001

内 容 简 介

本书依据国内外关于 \TeX 与 \LaTeX (尤其是当前最新版本的 $\text{\LaTeX 2}_{\epsilon}$)的资料,并结合编著者多年来使用该排版软件的经验,详细讲解了 $\text{\LaTeX 2}_{\epsilon}$ 这一功能强大的科技文献排版软件。本书既完整地介绍了中英文 $\text{\LaTeX 2}_{\epsilon}$ 所有排版命令,又对其编程功能做了讲述。

无论是初学者,还是经验丰富的 \TeX 用户,本书都是应用 $\text{\LaTeX 2}_{\epsilon}$ 排科技文章的最佳参考书。

图书在版编目(CIP)数据

$\text{\LaTeX 2}_{\epsilon}$ 科技排版指南/邓建松,彭冉冉,陈长松编著. —北京:科学出版社,2001.9

ISBN 7-03-009239-2

I .L… II .①邓… ②彭… ③陈… III . 排版-应用软件, \LaTeX
IV .TS803.23

中国版本图书馆 CIP 数据核字(2001)第 10090 号

科学出版社 出版

北京东黄城根北街16号

邮政编码:100717

<http://www.sciencep.com>

西源印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2001年9月第 一 版 开本:787×1092 1/16

2001年9月第一次印刷 印张:24

印数:1—3 000 字数:545 000

定价: 40.00 元

(如有印装质量问题,我社负责调换〈北燕〉)

前 言

编著者第一次接触 $\text{T}_{\text{E}}\text{X}$ 和 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 大约是 10 年前的事情了, 当时可用的资料只有一些油印的小册子。那时, 编著者还是一名本科生, 没有进行任何科研工作, 因此也就没有对它发生多大的兴趣。

6 年前编著者开始读研究生, 需要自己录入排版, 但是这时首先选择的排版软件是 Microsoft Word (版本 5.0), 因它具有“所见即所得”的良好性能。当时编著者用它录入了大量的文章, 甚至成本的教材。直到有一天, 编辑说用 Word 排版的数学公式太难看了, 这时才知道, 在文章的排版中, 要想得到标准而漂亮的数学公式, 并不是一件简单的事情。从这时起编著者才真正地开始学习和使用 $\text{T}_{\text{E}}\text{X}$ 和 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 。

从初学者的角度来看, $\text{T}_{\text{E}}\text{X}$ 和 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 实在是太难掌握了, 因为别的不说, 单是每开始排版一篇文章时, $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 文档开头的导言部分就使人满头雾水。而各种名称的 $\text{T}_{\text{E}}\text{X}$ 格式 (如 Plain $\text{T}_{\text{E}}\text{X}$, $\mathcal{A}_{\mathcal{M}}\mathcal{S}\text{-T}_{\text{E}}\text{X}$, $\text{emT}_{\text{E}}\text{X}$, $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$, $\mathcal{A}_{\mathcal{M}}\mathcal{S}\text{-L}_{\text{A}}\text{T}_{\text{E}}\text{X}$) 就更是令人眼花缭乱了。另外, 初学者在用 $\text{T}_{\text{E}}\text{X}$ 排版文章时, 经常会出错, 那些难以理解的出错信息, 也实在令人却步。

$\text{T}_{\text{E}}\text{X}$ 是一种文本处理系统, 它是美国 Stanford 大学的 Donald Knuth 于 1977 年 5 月开始着手设计的。几年后, Leslie Lamport 开发了 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$, 它以 $\text{T}_{\text{E}}\text{X}$ 作为自己的格式处理基础。 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 的诞生, 对 $\text{T}_{\text{E}}\text{X}$ 的普及起了相当重要的作用。发展到今天, $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 已逐步成为了主流, 因此本书中主要介绍的是 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$, 而且重点是它的最新版本 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ 。我们详细地介绍了如何应用 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ 进行科技文章和书籍的排版。

我们编写本书的主要目的就是帮助初学者快速掌握 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$, 但实际上即使对已经用过相当长时间的 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 用户, 本书也不失为一本内容齐全的参考书。在编写本书的时候, 我们参考了国内外出版的各种 $\text{T}_{\text{E}}\text{X}$ 和 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 使用手册和资料 (在参考文献中列出了这些资料的详细目录)。另外, 由于美国数学会推出的 $\mathcal{A}_{\mathcal{M}}\mathcal{S}\text{-L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 具有相当丰富的数学符号, 并且可以排出更复杂的数学公式, 因此我们在第五章中也同时介绍 $\mathcal{A}_{\mathcal{M}}\mathcal{S}\text{-L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 的数学排版功能。

全书共分四部分, 下面我们简要列出本书的主要内容。

第一部分为 **基本 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$** , 这一部分共包含两章, 其中第一章简要介绍 $\text{T}_{\text{E}}\text{X}$ 和 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 的历史, 并列出了编辑 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 文档的两种推荐编辑器 EditPlus 和 WinEdt。在第二章中对利用 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 排版文章进行快速简略的介绍, 主要列出了 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 命令的组成、各种字符的输入方法和长度的定义, 并给出了排版一篇完整文章的示例。这个示例中包含了 20 多个典型公式的样例, 这组样例对后面进一步学习公式的排版相当有帮助, 在阅读第五章时也不妨参考一下。

第二部分为 **高级 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$** , 详细完整地介绍了所有 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 排版功能。第三章介绍页面组织的各项功能, 其中包括对文档类的简要介绍, 列出了类的样式选项, 以及在相应的文档类中可以出现的章节命令及结构。接着介绍生成目录表、插图和表格清单的方法, 最后说明如何对文本中不恰当的地方进行细微调整, 并解释了控制断词的命令和方法。

第四章对文本排版的各种环境和声明进行了详细介绍。列举了改变字体大小和属性的命令,给出了排版居中和缩进文本块的方法。本章中不但介绍了 \LaTeX 预定义的各种列表环境,如`enumerate`、`itemize`和`description`等,而且给出了广义列表的语法,用这种列表可以构造出前一节的所有列表,而且还可以生成其他各种特色的列表环境。另外,还说明了在文章中如何使用定理结构、制表位和盒子命令。本章最后的几节介绍了显示源文本,添加注释和标记脚注与边注的方法。

第五章介绍的是数学公式的排版,当初之所以开发 \TeX 和 \LaTeX 系统,就是为了排版出标准而规范的数学公式。首先讲解了数学环境的构造,然后逐一说明了公式的组成部分,列出了获得各种数学符号的命令,并说明了公式中其他要素的构造方法,例如给公式加编号,多行公式的编辑等。在介绍前面内容的过程中,我们同时列出了 \LaTeX 中对 \LaTeX 数学排版增强的部分,特别地,我们在5.6节中介绍了 \LaTeX 为排版多行公式而增加的新环境和功能。本章的最后一节说明如何对数学公式进行精细调整。

第六章中讲解了 \LaTeX 中插图和表格的编辑方法。我们在前几节中介绍的是生成插图的命令,接着解释了生成表格的环境和参数,最后给出了 \LaTeX 文档中浮动对象的处理方法。

第七章中给出了在 \LaTeX 中定义命令和环境的方法。为此首先介绍了 \LaTeX 记数器和长度,然后详细介绍了在 \LaTeX 如何自定义无参数、有参数以及有可省参数的命令和环境。

第八章中介绍的则是比较复杂的功能,例如多文件的组合、正文中的引用、参考文献的组织、新字体选择框架的启用,以及书信和幻灯片的编辑等,另外还介绍了如何插入外部图形的方法。

在第三部分,我们介绍的是CCT中文 \LaTeX 系统,这一部分内容主要参考了CCT开发者提供的使用手册。首先在第九章介绍了各种中文 \TeX 系统,并说明了CCT中文 \LaTeX 的构成。在第十章则详细列举了CCT中文 \LaTeX 系统新增的命令和功能。

本书最后一部分是长达100多页的附录,内容包含了各种符号的列表,以及高级用户可能会感兴趣的编辑参考文献数据库的方法和 \LaTeX 程序设计命令,为了帮助用户在使用 \LaTeX 时应付各种出错和警告信息,我们也列出了针对各种信息的处理方法,在附录中还介绍了计算机现代字体,并在附录的最后给出了索引,以方便用户检索各种命令和概念。

为了方便 \LaTeX 爱好者交流经验,共享资料,编著者在1999年4月建立了 \TeX Guru站点,当前网址为<http://202.38.68.78/~texguru/>,因此编著者在这里要感谢中国科学技术大学数学系科学计算与计算机图形学实验室,如果没有该实验室的良好软硬件环境,本书是不可能这么快完成的。另外,科学出版社的杨波先生对本书的出版起了非常大的作用,作者在此向他表示衷心的感谢。

邓建松

2000年8月于中国科学技术大学

第一部分 基本 L^AT_EX



第一章 简介

在当今时代，计算机最通用的一个功能就是对文本的电子化处理，这一过程主要由如下四步组成：

1. 文本输入到计算机里，存贮供以后修改；
2. 把输入文本格式化，即用长度相同的行和特定尺寸的页排出来；
3. 在计算机的显示器上显示查看格式化后的结果；
4. 把最终的输出送到打印机上打印出来。

有很多文字处理系统（例如，MS Word）可以在一个软件包中同时实现这四方面的功能，因此用户也就意识不到上面的步骤划分。而且，第 3,4 步实际上是相同的，都是把格式化后的结果送到一个输出设备上。

T_EX 是一种文本格式化程序，它只进行第 2 步的处理。任何一种文本编辑器都可以进行第 1 步的操作，即输入和修改源文本。而字处理程序就不一定满足这里的要求，因为这种程序通常在文件中加入很多不可见的控制字符。对字处理程序而言，“所见即所得”是一个非常好的功能。但是另一方面，有的时候，我们更看重排版结果的优美与规范，以及系统的稳定性。因为谁也不想自己辛辛苦苦编辑出来的结果，由于系统崩溃而毁于一旦。

用编辑器生成的文件，若要用作格式化程序的输入，其中应该包含一些特殊命令，这些命令是用可以看到的普通文本表示的。从某种意义上来说，这种供格式化用的命令集合很像一种包装语言，它只是用来表示段落、章节等等从哪儿开始，而不是直接对文本进行格式化处理。而在用程序进行格式化的过程中，这些指令是如何被解释的，则要看所选用的版面设计格式而定。同样的文本，在不同的版式下可能形成完全不同的式样。

格式化程序的功能远不止这些。实际上 T_EX 也是一种有丰富功能的编程语言，有经验的用户可以用它编写代码，以增加某些功能。L^AT_EX 自身也就是一组复杂的宏集合。任何用户都可以通过编程，或者直接利用其他程序员已设计好的宏对 L^AT_EX 进行扩展。T_EX 和 L^AT_EX 的功能并不只限于包含在基本安装版本中的那些内容。

对于格式化软件而言，文本处理的最后一步是把结果送到输出设备上，这里的输出设备可以是打印机、计算机显示器，甚至文件。实现这种转化的软件称为驱动程序；它

把格式化程序已编码好的输出翻译成用户可以使用的某一设备上的特定指令。这也就是说，对每种类型的打印机，也就必须有相应的驱动程序。

1.1 T_EX 及其历史

在所有的可以排版科技著作的计算机格式化程序中，功能最强的就要属 Stanford 大学的 Donald E. Knuth 所设计的 T_EX 程序了，其名字是由在数学公式中经常用的希腊字母 $\tau\epsilon\chi$ 的大写形式组成。正是由于这个原因，T_EX 最后一个字母的发音并不是 [ks]，而是 [k]。

除了 T_EX 外，Knuth 还设计了另一个软件 METAFONT，用来生成各种字符字体。在标准的 T_EX 软件包中有 75 种不同设计尺寸的字体，而且每种字体有八种不同的放缩比例。所有这些字体都是用 METAFONT 程序生成的。为了满足其他应用的需要，还设计了其他字符字体，如古斯拉夫语和日语字母的字体，利用这些字体，就可以把相应文本按书籍质量要求排版。

1.1.1 T_EX 程序

最基本的 T_EX 程序由一些很初等的命令组成，它们可以完成简单的排版操作和程序设计功能。然而，在 T_EX 中还可以用这些初等命令定义一些复杂的高级命令。这样就可以利用低级的结构块，形成一个用户界面相当友好的环境。

当运行 T_EX 时，该程序首先读取格式文件（格式文件中包含各种以基本语言写成的高级命令，也包含分割单词的连字号安排模式），接着就处理源文件（源文件由要处理的真正文本，以及在格式文件中已有定义的各种命令组成）。

创建新格式也是一件需要由知识丰富的程序员来做的事情。把定义写到一个源文件中，这个文件接着被一个名叫 initex 的特殊版本的 T_EX 程序处理。它采用一种紧凑的方式存贮这些新格式，以利于通常的 T_EX 程序很快地读取。

1.1.2 Plain T_EX

Knuth 设计了一个名叫 Plain T_EX 的基本格式，以便与低层次的 T_EX 对应。这种格式是 T_EX 排版的相当基本的部分，以致于我们有时候根本分不清到底哪是真正的 T_EX 处理程序，哪是这个特殊的格式。大多数声称只使用 T_EX 的人，实际上指的是只用 Plain T_EX。

Plain T_EX 也是其他高级格式的基础，这些格式的广泛应用进一步加深了人们把 T_EX 和 Plain T_EX 认为是同一事物的错觉。

1.1.3 L^AT_EX

Plain T_EX 的重点还只是停留在如何排版的层次上，而不是从一位作者的角度来看问题。当然对 T_EX 深层功能的进一步发掘，需要相当高超的编程技巧。因此它的进一步应用就需要高级排版和程序设计人员的参加。

正是由于这种原因, 美国计算机学家 Leslie Lamport 开发了 L^AT_EX 格式, 这种格式提供了一组生成复杂文档所需要的更高级命令。利用这种格式, 即使使用者没有排版和程序设计的知识也可以充分发挥由 T_EX 所提供的强大功能, 能在几天, 甚至几小时内生成大量具有书籍印刷质量的结果。在生成复杂表格和数学公式方面, 这一点表现得尤为突出。

L^AT_EX 相对于其基础 Plain T_EX 而言, 更像一个包装语言。它可以在作者根本不知道所以然的情况下, 自动给出标题、章节、表格目录、交叉索引、公式编号、文献引用和浮动图表。版面布局信息包含在类文件中, 这些类文件并不是位于源文件中, 使用者不但可以直接套用这些布局, 还可以进行修改。

L^AT_EX 是在 20 世纪 80 年代出现的, 同其他软件一样, 它也周期性地更新和修订。

1.1.4 L^AT_EX 2_ε

由于 L^AT_EX 相当普及, 以及它在许多原本没有想像到的领域中的扩展, 再加上计算机技术的日新月异, 特别是价格低廉而功能强大的激光打印机的出现, 使得相当广泛的一类格式都冠以 L^AT_EX 的标签。为了尝试建立一个真正的改进标准, 在 1989 年 Leslie Lamport, Frank Mittelbach, Chris Rowley 和 Rainer Schöpf 创立了 L^AT_EX3 项目。他们的目标是建立一个最优的、有效的命令集合, 这些命令均来源于为了实现某一目的而设计的各种宏包。

正如项目名称所表明的, 它的目标就是得到 L^AT_EX 的一个新版本 3。朝向这个长期目标迈进的第一步就是在 1994 年中发行了 L^AT_EX 2_ε 并出版了 Lamport 基本手册第二版。L^AT_EX 2_ε 是在令人瞩目的 L^AT_EX3 出现之前的现行标准版本。

实际上, 在 L^AT_EX 2_ε 出现之前, 其处理字体的安装和选择的一些部分已经以新字体选择框架 (或 NFSS) 的形式公开了, 而且被许多组织和个人集成到其软件中。这种框架有两个版本, 但不幸的是, 这两个分别相应于 L^AT_EX2.09 和 L^AT_EX 2_ε 的版本并不兼容。后来以一种与 2.09 版本完全兼容的方式对 NFSS 进行了重新实现。

1.2 L^AT_EX 2_ε 的新内容

本节内容主要针对那些已相当熟悉 L^AT_EX2.09 的读者。下面简要列出了 L^AT_EX 2_ε 的新功能。如果读者是第一次接触 L^AT_EX, 完全可以略过该节内容。

1.2.1 类与宏包

L^AT_EX2.09 与 L^AT_EX 之间的一个最本质的差别就在于文档的第一条命令, 这条命令声明了文档的布局信息。

在 L^AT_EX2.09 中, 必须像下面这样来声明所需要的主样式, 这个样式同时带有一些选项:

```
\documentstyle[ifthen,12pt,titlepage]{article}
```

这里的主样式是 `article`，它保存在一个叫 `article.sty` 的文件中，而同时用 `12pt` 作为基本字体大小，`titlepage` 规定把标题单独放在一页上。但是由于主样式并不理解 `ifthen` 选项，所以会自动读入名为 `ifthen.sty` 的文件。这种处理未定义选项的方法是把 \LaTeX 的扩展或补充代码读入文档中的主要途径。类似于 `ifthen.sty` 的附加文件有很多，都可以用来做为样式选项（我们也称之为子样式）。

然而，在 \LaTeX 中这些补充选项与真正的内部选项之间是有显著差别的。因此现在主样式更名为类，而附加文件则被称为宏包。上面那个初始化声明就变为

```
\documentclass[12pt,titlepage]{article}
\usepackage{ifthen}
```

这里的布局信息包含在 `article.cls` 文件中，它可以处理 `12pt` 和 `titlepage` 选项。而 `ifthen.sty` 文件还像以前那样读取，但是它也可以有自己的内部选项。不仅如此，那些列在 `\documentclass` 命令中的选项，都被看作是全局的，因此对所有宏包都有作用（见 3.1.2 节）。

初始化声明 `\documentstyle` 在 $\text{\LaTeX}2_{\epsilon}$ 中仍可以使用，这时 $\text{\LaTeX}2_{\epsilon}$ 会切换到一个兼容模式，来模拟 $\text{\LaTeX}2.09$ 的行为。

为了帮助那些坚韧不拔的 \LaTeX 程序员更好地进行程序设计， $\text{\LaTeX}2_{\epsilon}$ 增加了许多新功能。它对选项的处理做了改进，还可以在宏包加进选项；增加了一些安全机制，以保证版本号的匹配；提供了更好的测试方法，以保证在读取其他文件时，如果该文件不存在，可以采取其他的措施。在附录 C 中对这些程序设计要素进行了描述。

1.2.2 字体管理

在 $\text{\LaTeX}2.09$ 中， \TeX 的计算机现代字体 (computer modern font) 被牢靠地固定在格式中。在人们喜欢用的字体也就那么几种的年代里，这不失为一种可行的方法。但到了今天，由于可用的字体数目繁多，尤其是 Post Script 打印机的出现，就非常需要设计一个有弹性的系统。新字体选择框架 (NFSS) 应运而生，并与 $\text{\LaTeX}2_{\epsilon}$ 结合为一体。要选择非计算机现代字体作为基本字体只需要一些很简单的重定义（见 8.5 节）。

NFSS 也改变了在文档内部引进字体的方法。 $\text{\LaTeX}2.09$ 继承了 \TeX 的字体命令，如 `\bf` (黑体)、`\it` (斜体) 都只能选择某一特定的字体。这些命令中只有字体大小维持不变。而在 NFSS 中，字体是用某种属性来描述的，可以彼此独立地进行选择。因此就有可能先选择黑体，然后选斜体，从而得到黑斜体，而这在 $\text{\LaTeX}2.09$ 中是行不通的。

$\text{\LaTeX}2_{\epsilon}$ 鼓励使用字体选择命令，不提倡使用字体声明。例如，为了强调某一文字，命令 `\emph{hello}` 就比声明 `{\em hello}` 要好。这样的命令对初学者来说，更符合逻辑，虽然习惯于用后者的经验丰富的 $\text{\LaTeX}2.09$ 用户可能持相反的看法。

在数学模式中，文本的字体是通过特殊的数学字母命令来选择的，而不是原来的字体声明。也就是说，在数学模式中不允许使用原来的 `\rm`、`\bf` 和 `\cal` 等声明，而代之以 `\mathrm`、`\mathbf` 和 `\mathcal` 等有参数的命令。

1.2.3 浮动对象的安排

在 L^AT_EX 2.09 中, 一个令人头痛的问题就是如何安排浮动对象 (图与表), 使它出现在人们最希望看到的地方。然而对浮动对象的安排有一套相当复杂的规则, 并不是所有的人都能很好地掌握。L^AT_EX 2_ε 提供了两种新的机制, 以控制这一过程, 其中一种不鼓励对浮动对象的安排, 另一种则鼓励这种安排。

通过使用 `\suppressfloats` 命令可以使当前页中没有浮动对象。作为选项, 可以使用参数 `t` 或 `b` 来只禁止浮动对象不出现在当前页面的顶部或底部。

另一方面, 有一个新的浮动位置指定符 `!`, 如果在浮动对象的定位参数中使用了这个符号, 那么它可以取消所在页对可以出现的浮动对象的数目与文本总量的限制。这也克服了通常会出现的一种页面不满的情形。而原来要解决这个问题, 就必须重定义 `\textfraction` 或其他浮动安排参数, 并进行无数次的调试才有可能成功。浮动位置指定符 `!` 的使用方法与其他参数一样。例如,

```
\begin{figure}[!]
```

现在为了使浮动对象与文本更好的分离开, 也可以在其顶部或底部画上定义好的标尺。在 6.7 节中对这些功能进行了介绍。

1.2.4 扩充的语法

与 2.09 版本相比, L^AT_EX 2_ε 对几条命令的语法进行了扩充。当然原来的语法仍然有效。

- `\newcommand`, `\renewcommand`, `\newenvironment` 和 `\renewenvironment` 命令用来定义新的命令和环境, 这些命令除了几个必须的参数外, 还可以包含一个可省略的参数 (7.3 节和 7.4 节)。
- 盒子命令 `\makebox`, `\framebox` 和 `\savebox` 在定义其实际尺寸时, 可以引用其自然的尺度。也就是说, 你可以用它来定义宽度为自然宽度两倍的盒子。见 4.7 节。
- 现在的 `\parbox` 命令和 `minipage` 环境除了需要定义水平宽度外, 还可以指定一个竖直高度。同时提供了一个新的内部安排参数来决定盒子中的文本是否需要被推向顶部或底部, 或者居中安置, 甚至可以伸展文本以填满整个盒子 (4.7.5 节)。
- 以前的 `\settowidth` 命令可以用来测量某些文本的宽度; 现在又补充了 `\settoheight` 和 `\settodepth` 命令 (7.2 节)。

1.2.5 与 L^AT_EX 2.09 的兼容性

为了使 L^AT_EX 2_ε 与 L^AT_EX 2.09 尽可能地保持兼容性, 系统设计人员已做了各种各样的努力。这样就可以使根据原来版本标准写的文档, 在新的版本下可以得到相同的结果。同时宏包中绝大多数样式 (或子样式) 选项具有与以前相同的功能, 而不需任何修正。

这只是最理想的情形。实际上, 两种版本之间一定存在着不兼容之处。那么用户在哪些情形中会遇到这种不兼容现象呢?

- 如果只使用高级命令, 在命令中不包含 `@` 字符, 那么兼容性是 100%。
- 如果使用了内部命令, 就可能会导致问题, 特别是用到了盒子命令或者输出程序。

- 如果使用了内部字体控制命令，特别是那些来自于 `lfonts.tex` 文件的命令，就很有可能会出现問題。然而，应该指出的是，即使这样，也比 NFSS 的第一个版本与第二个版本之间的不兼容性小。

据目前的经验，我们发现，即使把盒子命令和输出程序算在内，也只有很少几个宏包不能在 \LaTeX 2_ϵ 下运行。我们遇到的最糟糕情形是，在 \LaTeX 2.09 下调入一个支持文件，而这个文件在 \LaTeX 2_ϵ 下却不一定存在。如果该文件被强行调入，就会出现严重错误的信息。

1.2.6 版本的判别

如果你不知道在你所用的系统中是否安装了 \LaTeX 2_ϵ ，有许多方法可以检查。在任一 \LaTeX 作业开始处，都会在显示器和抄本文件 (transcript file) 中列出格式的名字和日期：

如果用的是 \LaTeX 2.09 ，信息为 `LaTeX Version 2.09 <25 March 1992>`，

如果用的是 \LaTeX 2_ϵ ，信息为 `LaTeX2e <1996/12/01>`，

当然你的日期可以与这里的不同。

如果不喜欢这种方法，还可以尝试处理一个文件，第一行用

```
\documentclass.
```

如果你得到如下的错误信息：

```
! Undefined command sequence
```

```
\documentclass
```

那么你用的就不是 \LaTeX 2_ϵ 。

1.2.7 更新 \LaTeX 2_ϵ

按计划在每年的 6 月和 12 月要对 \LaTeX 进行更新，不仅要改进内部编码，还要增加一些额外的功能。这就是说，随着时间的推移，仅仅只说一个文档是 \LaTeX 2_ϵ 格式的还不够，不但要指明必要的版本，还要加上格式发行的日期。

为此我们需要在文档文件中声明一个最早的可以处理文档所用全部功能的版本。在靠近文档开头的地方，加上一条鉴别命令 (C.2.1 节)。如果用的是第一个 \LaTeX 2_ϵ 版本，那就应该加上

```
\NeedsTeXFormat{LaTeX2e}[1994/06/01]
```

这里的日期必须是数字，格式为年 / 月 / 日，年月日中间用斜线分开，且须加上必要的零。如果有这一命令的文件被更早的 \LaTeX 版本处理，就会显示一条警告信息。

1.3 如何使用本书

本书是教科书与参考手册的混合体。它解释了所有的 \LaTeX 以及 \CCT emTeX 的基本组成，特别是那些新标准的 \LaTeX 2_ϵ 中的部分。本书适用于那些经验很少，甚至没有计算机使用经验的用户。

书中的内容是逐步加深的，在第一部分与第二部分中有些内容是重复的。因此读者在刚开始学习的时候，没有必要强求掌握新出现的所有概念。

1.3.1 字体约定

为了更好地理解本书所述内容，我们建议读者要首先熟悉一下从 2.2~2.5 节开始出现的那种描述命令或环境语法的方式。

在描述命令语法时，对那些必须精确照原样输入的部分用打字机字体（如 `\begin`）表示，而那些可以改变的部分或者文本自身，则用斜体（相应于英文符号）或楷体（相应于中文）表示。例如，宏包 `graphpap` 中生成格纸的命令就用如下方式陈述：

`\graphpaper[数](x,y)(lx,ly)`

用打字机字体表示的部分是不能省略的，而 `数` 表示必须在此处加上格线间隔的单位数。而 `x`, `y`, `lx`, `ly` 分别表示格式的左下角坐标以及格式的尺寸。

在正文中，有时为了表示强调，我们也用楷体表示特定的术语或规则，例如：“公式中有时会包含一串点 \dots ，表示 等等”。

1.4 L^AT_EX 的编辑

我们在前面已说过，编辑 L^AT_EX 可以采用任何文本编辑器，但前提条件是这个编辑器生成的文件必须是没有格式的，即它不会向文件中增加任何不可见的控制字符。目前非常流行的 Microsoft Word 是一个非常优秀的编辑器，用它排版没有数学公式或者只有少量公式的普通文章或办公信函，是非常便捷的，因为它具有“所见即所得”的强大优势。但是这个编辑器并不满足我们这里的要求。

在微机上，如果所用操作系统为 DOS 或 Windows，那么最经常用的文本编辑器是 Edit。但是采用这个编辑器时，我们不得不频繁进出编辑器，以编译和查看排版结果。因此我们期望能有一个集成式的编辑环境。我们要求这个环境，一方面具有丰富的编辑功能，另一方面，不用退出编辑器，就可以马上知道排版结果。EditPlus 和 Winedt 是我们的首选。

1.4.1 EditPlus

EditPlus 文本编辑器是 ES Computing 公司提供的 32 位的 Windows 程序，可以用它替换 Windows 附件中 Notepad 编辑器，它也为网页的编辑与设计提供了许多强有力的工具。

这个编辑器可以按照系统预定的语法文件或者用户自定义的语法文件，用彩色显示出某种类型文件中的保留词或命令。例如，若正在编辑 C++ 源程序，那么当输入完 `float` 的最后一个字母时，这个单词就会以彩色显示，从而提示你的输入是正确的。

另外这个编辑器还允许用户利用外部程序设置一些工具，而且有自动补足的功能。正是由于这个编辑器具有这些强大功能，我们才选择它做为 L^AT_EX 的编辑工具。首先我们创建出相应于 L^AT_EX 类型的语法文件和自动补足文件，然后对 EditPlus 进行一系列的配置或

直接修改注册表文件(配置或修改的具体过程以及所需的相关文件请见 <http://202.38.68.78/~texguru/> 站点)。在做了上述配置或修改以后, 现在编辑 \LaTeX 文件时, 就有很多便利的地方, 一方面, 所有的 \LaTeX 命令在输入完毕后, 都会自动变色(要保证在命令的前后与其他命令之间有适当的分隔, 如空格), 甚至 \TeX , \LaTeX 和 \AA\MS-\LaTeX 的命令所具有的颜色是不同的, 从而提示你包含必需的宏包文件。在使用环境时, 编辑器会自动补足 $\text{\end{...}}$ 部分。例如, 当输入完

```
\begin{equation
```

后, 我们不要输入另一半 \ , 只要按空格, 那么你就会得到

```
\begin{equation}
```

```
\label{}
```

```
\end{equation}
```

光标自动停留在 $\text{\label{}}$ 的大括号内, 让你输入一个关键词, 以供引用。由于我们使用 \equation 环境, 就是为对这个公式进行引用, 否则用的就是 $\text{\$...\$}$ 环境了。注意, 这里千万不要在输入 \equation 后的 \ 再按空格, 那样是不会有这种自动补足效果的。

另外, 我们还可以利用 EditPlus 自动生成 \LaTeX 模板, 因为初学者对 \LaTeX 导言部分的组成是非常头疼的。利用模板, 就可以很容易地着手编辑正文。下面就是我们自己创建的 \LaTeX 文件模板。

```
\documentclass[11pt]{article}
%\documentclass[11pt]{book}
%\usepackage{graphicx} %We can use any other package if necessary
%\usepackage{amsmath} % If you want to use AMS-LaTeX, comment out these
%\usepackage{amssymb} % two commands.
%\usepackage{chemsym}

\setlength{\parindent}{12pt}
\setlength{\parskip}{3pt plus1pt minus2pt}
\setlength{\baselineskip}{20pt plus2pt minus1pt}
\setlength{\textheight}{21true cm}
\setlength{\textwidth}{14.5true cm}

\title{Thesis}
\author{Deng Jiansong\text{\texguru@263.net}}

\begin{document}%

%Create title of papers.
```

```

\maketitle

%Edit the contents as you want.
%If you want to insert some PostScript pictures, use command:
%\includegraphics{xxx.eps}

\end{document}%

```

只要在 EditPlus 文件中, 从菜单上选择 New, 然后选你要创建的文件类型 (CCTeMTeX 或 LaTeX), 就可以得到不同类型的模板。把这个文件保存成后缀为 .tex(相应于英文 L^AT_EX) 或 .ctx(相应于 CCT 中文 L^AT_EX) 的文件, 然后就可以输入和编译。在上述模板中, 我们在许多行前面加上了注释符号 %, 这样在实际使用时, 就可以通过去掉某些行前面的注释符号, 或者在某些行前面加上注释符号, 而改变所使用的缺省设置。例如, 可以把

```

\documentclass[11pt]{article}
%\documentclass[11pt]{book}

```

变为

```

%\documentclass[11pt]{article}
\documentclass[11pt]{book}

```

这样文档使用的就是 book.cls 类, 而不是 article.cls 类。

1.4.2 WinEdt

WinEdt32 是由 Aleksander Simonic 开发的一个快速、强大的编辑器, 它具有相当高的弹性, 可以在 Win95, Win98 和 WinNT 等系统中使用。该系统是用 Borland Delphi 开发的 32 位应用程序。

可以用 WinEdt 编辑相当庞大的文本文件。可以同时打开多个文件, 遵从通常多文档界面 (MDI) 的约定, 利用窗口菜单或 "Ctrl+Tab" 按键在文件中来回切换。除了一些标准的编辑功能外, WinEdt32 还支持:

- 块 / 列选择和操作, 关键词高亮显示, 拼写检查并显示出拼写有误的单词, 定界符自动匹配, 恢复桌面, 宏功能, 用户自定义图形界面,
- WinEdt 可以搜集文档中的特定信息 (例如目录表或者标签), 以便于快速移动和索引。

WinEdt 的弹性表现在

- * 用户可完全自定义菜单 (包括快捷键) 的组成;
- * 可完全自定义工具条;
- * 用户可自定义与不同上下文有关的弹出菜单, 当单击鼠标右键时会出现这些菜单。

我们可以对 WinEdt 进行配置, 以适用于不同的需要。不过, 它特别适合于编写 T_EX 文档, 特别地, 它提供了一组模板, 要输入 T_EX 和 L^AT_EX 甚至 \mathcal{A} S-L^AT_EX 符号, 只要在这个模板的对应符号处单击一下就可以输入该符号, 这大大简少了记忆符号名称的工

作量。不过，相对于 EditPlus 而言，我们至今还没有找到显示行号的功能，这也不能不说是一种遗憾。在 WinEdt 中我们可以安装 DOS 和 Windows 程序，然后只要利用菜单命令或者工具条按钮就可以激活这个程序。而且，WinEdt 可以把当前打开的文件做为参数传递给这个程序。在定义了一些工具（例如 $\text{T}_{\text{E}}\text{X}$, $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$, $\text{B}_{\text{I}}\text{B}_{\text{T}}\text{E}_{\text{X}}$ 等）后，与编写 $\text{T}_{\text{E}}\text{X}$ 文档有关的绝大多数任务都只需要单击一个工具条按钮或者选择一条菜单命令就可以完成。

WinEdt 起初是与 $\text{Mik}_{\text{T}}\text{E}_{\text{X}}$ 一同使用的，为了使其支持中文 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ ，我们也可以对其进行一些配置，具体详情见 $\text{T}_{\text{E}}\text{X}$ Guru 主页 (<http://202.38.68.78/~texguru/>)。

第二章 L^AT_EX排版入门

L^AT_EX 是目前国际上流行的排版软件,它特别适合于科技文章和书籍的编辑和排版。与目前流行的 WPS 和 Word 软件相比,它在字符质量、排版功能和数学公式的处理方面均胜一筹。目前,许多著名的期刊,如 SIAM, AMS, Phy. Rev. 等都接受 L^AT_EX 格式的稿件,甚至有许多杂志、期刊有自己的类与宏包,以便对文章做出满意的排版。

在本章简单介绍 L^AT_EX 排版的基本概念,并演示一些结构的使用方法。

2.1 L^AT_EX 文件的基础知识

2.1.1 文本与命令

所有的文本文件都由字符组成,几个字符放在一起组成单词,由多个单词组成句子,句子再组成段落,而段落可以做为更大单位(如章节)的一部分。

单词由一个或多个字符组成,由空格或回车终止。T_EX 把空格和回车都做为单词的结束符,而单词间的空格多少是无关紧要的,即单词的间隔与空格的数目无关。

段落是由一个或多个空行分隔的,同样,段落间隔与空行的数目无关。T_EX 把段落中的所有单词当做一个单词长串来处理,选择单词间隔,以使得尽可能地均匀,而且每一行都要向左或向右调整(对齐)。行的断开是自动的,与文本的输入基本无关。

行的间隔与所选的字体尺寸有关。段落是以首行的缩进或者行间隔的增大为标志的。在必要的时候,段和行的间隔都会发生细微的变化。T_EX(和 L^AT_EX)通过调整这些间隔,使得一页的顶部和底部位于需要的位置。可以在文档内改变这个位置。当断行结束后,会自动地进行分页。

对于最简单的情形,即文本文件中只有输入文本,T_EX 就会用标准的宽度和高度处理这一文本。也就是说,把要排版的文本均匀分成行、段、页。

然而,通常的 L^AT_EX 文档不仅仅是由要处理的文本组成,还要包含规定处理文本方式的命令。因此就有必要提供一种方法,区分开这些命令和文本。命令既可以由不能做为普通文本的单个字符组成,也可以由单词组成,它们前面都要加上一个特殊的字符,即反斜杠 \。

2.1.2 L^AT_EX 文档的结构

在每个 L^AT_EX 文档中,都必须有导言(preamble)与正文(body)两部分。

导言是一组命令的集合,它指定处理后面文本的全局参数,如页面格式,文本的高度和宽度,输出页的页码、页眉与页脚的组成。即使是在最简单的情形中,导言也必须包含命令 \documentclass,以指定文档的全局处理类型。这通常也是导言中的第一条命令。

如果在导言中再没有其他命令，L^AT_EX 就会为行宽、页边、段落间隔、页面高度与宽度以及其他度量选择默认值。在以前的 L^AT_EX 版本中，这些设置采取的是美国标准。对于欧洲用户所用的程序，存在内建的选项，使文本的高度和宽度符合 A4 纸标准。现在已经有了与语言相关的宏包，可以把 ‘Chapter’ 和 ‘Abstract’ 的标题翻译成相应语言。而将在第十章中讲述的 CCT emT_EX 则是通过对类文件进行修改而做到这一点的。

导言用 `\begin{document}` 表示结束。在第 8 页给出的 L^AT_EX 文件示例模板中，`\begin{document}` 命令前面的部分，就是一个典型的导言结构。在 `\begin{document}` 命令后面的所有内容都被看成正文。它由文本混杂命令组成。与导言的内容相比，这些命令只有局部的作用，即它们只适用于一定的范围，如缩进、公式、对字体的暂时改变，等等。正文是用 `\end{document}` 结束的。这通常也是文件的结束。

一个 L^AT_EX 文件的通常语法如下：

```
\documentclass[选项]{类}
其他全局命令和定义
\begin{document}
文本与只有局部作用的命令的混合
\end{document}
```

我们将在 3.1.1 节中介绍所有可以出现在 `\documentclass` 命令中的选项和类。初学者可以照抄我们在第 8 页上给出的示例。随着学习进程的推进，很快就会明白所有命令的功能。

如果所用的是 L^AT_EX 2.09，或者与之兼容，但在 L^AT_EX 2_ε 之前出现的版本，就必须用 `\documentstyle` 命令取代 `\documentclass` 命令。L^AT_EX 2.09 的一般性语法应该是：

```
\documentstyle[选项]{类}
.....
\begin{document}
.....
\end{document}
```

2.1.3 L^AT_EX 的处理模式

在处理过程中，L^AT_EX 总是处于下面三种模式之一：

1. 段落模式，
2. 数学模式，
3. 从左到右 (LR) 模式。

段落模式也就是正常的处理模式，这时 L^AT_EX 把输入文本做为一串要被 (自动) 断成行、段落和页的单词和句子。

当遇到特定命令，表示下面的文本代表公式时，L^AT_EX 就会切换进入数学模式，在公式中空格被忽略。例如，`ln` 和 `l n` 都解释成 l 与 n 的乘积 ln 。当遇到相应的表示公式结束的命令时，L^AT_EX 就会切换回段落模式。

LR 模式类似于段落模式, 这时 L^AT_EX 从左到右处理输入文本, 把它们做为一组不能被断行的单词来看待。例如, 当普通文本被嵌入到公式中时, 或者用命令 `\mbox{短文本}` 来强迫 短文本 位于同一行时, L^AT_EX 就处于 LR 模式。

理解与识别处理模式是相当重要的, 因为有些命令只能用在特定的模式中, 或者在不同的模式中有不同的作用。今后我们将把段落模式和 LR 模式合称为文本模式, 以表明它们的性质是一致的。但我们还是要把它们与数学模式区分开, 因为这两者之间的差别是很大的。

2.1.4 生成 L^AT_EX 文档

从输入文本到在打印机上得到的 L^AT_EX 排版结果, 是由三步组成的。首先利用计算机的编辑器创建 (或修改) 文本文件。这个文本文件由实际的文本混杂 L^AT_EX 命令组成。文本文件的全名由基本名加上扩展名 `.tex` 组成 (如 `sample.tex`)。如果用的是 CCT 中文 L^AT_EX, 那么后缀就应当为 `.ctx`, 但我们要用另外的程序把它翻译成 `.tex` 文件。计算机操作系统通常对文件名的选取有一些限制, 如基本名和扩展名可以拥有的最多字符数, 或者哪些字母不可以使用, 等等。例如, 如果基本名只限于不超过 8 个字符, 那么就可以使用 `finances.tex` 做文件名, 而 `financial.tex` 是不可以的。

然后用 L^AT_EX 处理文本文件。在 1.1.1 节中已做了介绍, 这时用 L^AT_EX 格式执行 T_EX 程序。在大多数安装版本中, 对此都有一个特定的缩略命令, 即 `latex2e`, 只要在该命令后输入文本文件的名称就可以了, 不必带后缀 `.tex`。例如, 如果文本文件的名称是 `sample.tex`, 那么应该用下面的调用来启动 L^AT_EX 处理程序:

```
latex2e sample
```

在这一处理过程中, 终端监视器会显示页号以及可能会有的错误和警告消息。附录 D 介绍了这些错误消息及其后果。当 L^AT_EX 结束了这一处理过程后, 它会生成一个新的文件, 其基本名不变, 后缀为 `.dvi`。在上面这个例子中, 得到的新文件就是 `sample.dvi`。

新生成的 `.dvi` (Device-Independent 的缩写, 表示与设备无关) 文件由格式化后的文本以及与所需要的字体有关的信息组成, 但是它与所用的打印机的指令无关。这种与设备无关的文件也叫做元文件 (metafile)。

2.1.5 浏览与打印结果

为了查看处理结果, 可以利用系统提供的屏幕浏览程序。目前 DOS 下的 L^AT_EX 系统中浏览程序一般名为 `view.bat`。而目前比较好的浏览程序是中国科学院计算数学与科学工程计算所的张林波提供的 `cctwin16` 和 `cctwin32` 程序。笔者在实际中用的就是 EditPlus 结合这个浏览程序进行 L^AT_EX 文件处理的。该程序可以查看中文 L^AT_EX 生成的结果。如果正确地对 EditPlus 进行了相关配置 (见 1.4.1 节), 那么在编译完文档文件 (按 `Ctrl+1`) 后, 只要按 `Ctrl+2` 就可以启动 `cctwin` 程序, 查看 `.dvi` 文件。而且如果接着在修改后重新编译了文档文件, 只要在 Windows 中切换 (按 `Alt+Tab`) 到 `cctwin` 程序中, 其中所显示的结果会自动刷新。

最后, 在 `.dvi` 元文件中的信息要被转化成可以在指定打印机上输出的形式, 这一过

程是由打印机驱动程序完成的, 这个程序的实现细节与打印机有关。目前在 CCT emT_EX 中提供了下述驱动程序:

24 针打印机驱动程序:	DVI24P.EXE (也用于驱动 Canon 喷墨打印机)
HP 激光打印机驱动程序:	DVILJP.EXE (适用于 HP LaserJet+ 激光打印机)
北佳激光打印机驱动程序:	DVIPECAN.EXE (适用于北佳视频卡)
喷墨打印机驱动程序:	DVIDJ500.EXE (适用于 HP DeskJet 系列喷墨打印机)

注意上述驱动程序只有 DOS 版本, 因此应在 DOS 命令行中调用。对驱动程序的调用, 类似于 L^AT_EX 程序的调用, 也只需要使用文件的基本名, 不必加上扩展名 .dvi。打印机驱动程序处理完这个 .dvi 文件后, 就又会生成另一个基本名相同的文件, 而扩展名与打印机有关, 或者直接送到打印机上输出结果。

例如, 如果输出要送到激光打印机上, 可以如下调用驱动程序:

```
dviljp sample
```

这样, 就会出现有如下几项的菜单窗口:

```
CCT Device Driver for HP LaserJet+, V5.13Z (Jan 16 1998)
      DVI file name  SAMPLE.DVI
      Reduction factor [Z]  1
      Font resolution (DPI) [R]  300
      Magnification [M]
      X margin:Y margin [B]  0cm:0cm
      X offset:Y offset [W]  Center:Center
      PK font path [P]  C:\EMTEX\PIXEL\DPI$d
      Chinese font path [I]  C:\EMTEX\HZFONTS
      Number of copies [C]  1
      Output format [F]  1x1
      Range of page numbers [N]
      Parity of page numbers [U]  Process all page numbers
      Page numbering [V]  TeX page numbers (\count0)
      Output orientation [O]  Portrait
      Reflection mode [Q]  Off
      Output device name [L]  LPT1
      Printing area (x0:x1:y0:y1) [D]  0cm:200mm:0cm:280mm
      Left or Right arrows to change current item, <Enter> to start
```

可以使用上下方向键在不同项之间移动, 也可以直接输入每项中间放在中括号内的字母, 以移动该项, 然后按左方向键或右方向键进入修改状态。修改完后按回车可退出修改状态。最后按回车就可以开始打印。另外, 也可以在 cctwin 程序中直接打印所查看的 .dvi 文件。

2.2 命令的名称与参数

字符 `# $ % ^ _ ~ % {}` 表示特殊的命令，它们的含义后面会讲到。如果要以文本的形式打印这些字符，就必须在它们前面加上 `\` (反斜杠)。

大约有二十条命令是由 `\` 后接另一个非字母字符组成的，即该命令只由两个字符构成。在这种类型的命令中，`\$` 表示暂时取消命令 `$` 的含义，而以文本的方式打印出这个字符。

绝大多数的命令是由 `\` 后接一个或多个字母组成的，第一个非字母字符表示命令名称的结束。许多命令可以有参数或变量来扩展相应的功能。参数有可能是可以省略的，也就是说可以给出它们特定值，也可以不给出。也有的参数是不能省略的，即至少要给出一个合法的值。这些命令的语法是

`\命令名[可省参数]{不可省略的参数}`

这里可省参数放在中括号 `[]` 里，而不可省略参数放在大括号 `{}` 里。一条命令可以有几个可省略参数，这时就必须按指定顺序放在各自的中括号内。如果没有给出可省略参数，中括号可以不写。在命令名称和参数之间可以有任意数目的空格。

对于可省略参数，有可能出现两个问题。一个是指如果没有使用任何可省略参数，而接下来的文本中第一个字符是 `[`，另一个情形是可省略参数的文本中包含 `]`，这时 `LaTeX` 就会误解参数。为此我们需要用大括号来保护中括号，例如 `{[}` 和 `}]`。

有的命令有几个不可省略的参数，那么就要把每一个参数都放在一对大括号内，而且要完全按照命令描述中的顺序排列。例如，

`\rule[提升高度]{宽度}{高度}`

就会生成一个大小为指定宽度和高度的实心矩形，并且相对于当前基线向上提升一定的高度。一个宽 10mm，高 3mm，且恰好位于基线上的矩形的生成命令为 `\rule{10mm}{3mm}`。在这里并没有给出可省参数。参数必须按描述中的顺序出现，不能交换。

有些命令有两种形式，一种是标准形式，而另一种就是所谓的 `*`-形式。后者的标志是命令名称以 `*` 号结尾，这个 `*` 号在相应于可省参数和不可省参数的成对的 `[]` 和 `{}` 之前。标准形式与 `*`-形式之间的区别，我们会在每条命令中加以解释。

命令名称是在第一个非字母字符之前结束的。如果命令后接可省参数或不可省参数，那么命令名称是在 `[` 或 `{` 之前结束的，因为这两个字符是非字母字符。然而有很多命令没有参数，只由名称组成，例如生成 `LaTeX` 标志的命令 `\LaTeX`。如果这样的命令后面接的是一个标点符号 (例如逗号或句号)，这显然表示命令的结束。如果它后接一个普通单词，那么命令名称与接下来单词间的空格被当做命令结束符：`\LaTeX logo` 生成的结果是 `LaTeXlogo`，也就是说，这里的空格只当做命令的结束符，而不当做两个单词的间隔。这是空格处理中的一个特殊规则，在 2.6.1 节中介绍。

为了在只由名称组成的命令之后插入一个空格，那就需要在命令后使用一个空结构 `{ }` 或空格命令 (`\` 加空格)。因此生成 'The `LaTeX` logo' 的正确方法是输入 `The \LaTeX{} logo` 或 `The \LaTeX\ logo`。如不这样做，也可以把命令放在大括号内，如 `The {\LaTeX} logo`，这样也会得到有空格的正确输出：'The `LaTeX` logo'。顺带说一下，用 `\TeX` 和 `\LaTeXe`

可以得到 T_EX 和 L^AT_EX 2_ε 标志。

2.3 环 境

环境是由命令 `\begin{环境}` 来初始化的, 最后用 `\end{环境}` 结束。至于 环境 可以取哪些值, 我们将在有关命令的章节中加以介绍。

环境的作用就是根据环境参数, 把其中的文本进行不同的处理。有可能 (暂时地) 改变某一处理特征, 如缩进, 行宽, 字样等等。这些改变只在该环境内有作用。例如, 对于 `quote` 环境

前面的文本

```
\begin{quote}
```

文本 1 `\small` 文本 2 `\bfseries` 文本 3

```
\end{quote}
```

后续文本

与前面和后续文本的左边和右边页边相比, 环境中的页边界要增大。具体到这个例子, 文本1、文本2和文本3的页边界都会增大。在文本1后面是命令 `\small`, 这样就把后面文本变成小字样。在文本2后又有一命令 `\bfseries`, 它切换到黑体字样。而所有这些命令的作用到 `\end{quote}` 时也就结束了。利用上面的环境可以得到如下结果:

在 `quote` 环境中的三部分文本都相比于前面和后面的文本进行了缩进。文本1是以正常字样出现的, 这同环境外是一样的。而文本2和文本3则是小号字样, 而且文本3还是黑体。

当 `quote` 环境结束时, 接下来的文本字样同以前的相同。

很多命令的名称与环境名称相同。在这种情况下, 所用的环境名称没有前缀 `\`。例如, `\em` 命令把当前字体切换为斜体, 以表示强调, 而相应的 `\begin{em}` 环境则把后续的所有文本变为斜体, 直到 `\end{em}` 为止。

一个没有名称的环境可以用一对大括号 `{...}` 来表示。其中所有命令的作用当遇到右大括号时也就结束了。

用户也可以创建另外的环境, 详情见 7.4 节的描述。

2.4 声 明

声明就是一种命令, 它并不显示任何文本, 而是改变某一参数或命令的取值或含义。声明一旦出现, 就马上发挥作用, 直到遇到同一类型的另一个声明为止。然而, 如果声明是位于一个环境或 `{...}` 中, 那它的作用也就只能延续到相应的 `\end` 命令或右大括号为止。在前一节中提到的命令 `\em`, `\bfseries` 和 `\small` 都是没有直接输出, 而只是改变当前字样的声明。有一些声明与参数有关, 例如 `\setlength` 命令会给长度参数赋值 (见 2.5 节和 7.2 节)。

下面给出几个例子:

```
{\bfseries This text appears in bold face}
```

这里的 `\bfseries` 声明改变了字样, 结果为: **This text appears in bold face.**
这个声明的作用在遇到右大括号 `}` 时结束。

```
\setlength{\parindent}{0.5cm}
```

把段落中第一行的缩进设为 0.5cm。当遇到下一个 `\setlength{\parindent}` 命令, 或者遇到结束当前最内层环境的 `\end` 命令时, 这个声明的作用就会终止。

```
\pagenumbering{roman}
```

用罗马数字显示页码。

有些声明的作用是全局性的, 如上面的最后那个例子, 也就是说, 即使当前环境结束, 它仍然具有作用。下面的声明都具有这种性质, 稍后给出它们的含义:

<code>\newcounter</code>	<code>\pagenumbering</code>	<code>\newlength</code>
<code>\setcounter</code>	<code>\thispagestyle</code>	<code>\newsavebox</code>
<code>\addtocounter</code>		

用这些命令给出的声明, 也是马上起作用, 而且直到被同类型的一个新声明覆盖其含义为止。在上面最后那个例子中, 只有当遇到 `\pagenumbering{arabic}` 这样的命令时, 才不会继续以罗马数字显示页码。

2.5 长 度

2.5.1 固定长度

长度是由前面可能有符号 (+ 或 -) 的浮点数, 后接一个必需的尺寸单位组成。下面是可允许的单位及缩写名称:

<code>cm</code>	厘米,
<code>mm</code>	毫米,
<code>in</code>	英寸 ($1\text{in} = 2.54\text{cm}$),
<code>pt</code>	点 ($1\text{in} = 72.27\text{pt}$),
<code>bp</code>	大点 ($1\text{in} = 72\text{bp}$),
<code>pc</code>	pica ($1\text{pc} = 12\text{pt}$),
<code>dd</code>	didôt 点 ($1157\text{dd} = 1238\text{pt}$),
<code>cc</code>	cicero ($1\text{cc} = 12\text{dd}$),
<code>em</code>	与字体相关的尺寸, 表示大写字母 M 的宽度,
<code>ex</code>	也是与字体相关的尺寸, 表示字母 x 的高度。

在 $\text{T}_{\text{E}}\text{X}$ 和 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 中的浮点数, 既可以采用英国方式, 用句号表示小数点, 也可以采取欧洲方式, 用逗号表示小数点, 因此 12.5cm 和 $12,5\text{cm}$ 都可以使用。

注意 0 不是一个合法的长度, 因为其没有长度单位。要给出一个零长度, 必须用 `0pt` 或 `0cm` 等等。

要想给长度参数赋值, 可以用 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 命令 `\setlength`, 我们将在 7.2 节中介绍这条命令以及其他所有处理长度的命令。 `\setlength` 的语法是:

```
\setlength{\长度命令}{已定义的长度}
```

例如，一行的长度是由一个叫 `\textwidth` 的参数定义的，它通常根据样式和字体尺寸来取默认值。要想把行宽设为 12.5cm，可以用：

```
\setlength{\textwidth}{12.5cm}
```

2.5.2 弹性长度

有些参数的值为弹性长度，即这个长度可以伸展或收缩给定的量。弹性长度的语法是：

正常值 plus 伸展值 minus 收缩值

这里的 正常值、伸展值和收缩值 都是长度。例如：

```
\setlength{\parskip}{1ex plus0.5ex minus0.2ex}
```

其含义为：两段间的行距（称为 `\parskip`）等于当前字体中 x 的高度，但是该行距可以伸长到给定长度的 1.5 倍，或者收缩到它的 0.8 倍。

有一个特殊的弹性长度是 `\filll`。其正常长度是零，但可以伸展到任何长度。

2.6 特殊字符

2.6.1 空格与回车

空格或空白字符具有与通常字符不太一样的性质，对此我们在前面 2.2 节中已讲到了一些。在处理的过程中，空格是用弹性长度（2.5.2 节）代替的，这样可以使得每一行恰好达到指定的宽度。因此，如果你不知道下面的规则，就可能发现一些古怪的现象：

- 一个空格同一千个空格是一样的，实际接受的只是第一个空格；
- 在一行开头的空格是被忽略不计的；
- 一个回车（开始新行）是当做空格对待的，而连续两个或两个以上的回车认为是段落结束。

根据这些规则，我们可以在单词间或一行的开头插入随意多的空格，从而使源文件更容易阅读，也可以把一个单词恰好放在一行的末尾，它和后一个单词之间没有空格，因为这里的回车就起着空格的作用。为了使在一般情形中会消失的空格必须出现，那就要用 `_`（后接一个空格键，这里用符号 `␣` 表示）命令。

面有的时候又有必要避免由于开始一个新行而出现不必要的空格。在这种情形中，该行必须在回车键前插入注释符号 `%`。

连续两个回车键得到一个空行，这就表示开始下一个段落。同空白字符一样，一个空行与一千个空行是完全一样的。

2.6.2 引号

在书籍印刷中用到的引号并不是在打字机上可以找到的引号（"），而是在开始和结束时要用不同的符号，如‘单引号’和“双引号”。单引号是用 ‘ 和 ’ 生成的，而双引号是用两个相应的字符生成的：‘ ‘ 得到 “，而 ’ ’ 则得到 ”。另外，由键盘直接输入 " 会生成右双引号。注意这里的 ‘ 位于普通键盘上主要按键部分的最左上角，与 ~ 在同一个按键上。

2.6.3 连字符与破折号

在书籍印刷中, 对应于打字机上为 - 的符号有各种不同的长度, 它们分别是 -, -, —。其中最短的是连字符 (haphen), 用在诸如 father-in-law 这样的复合单词中, 以及在一行结尾处的单词分割中 (这种断词是 T_EX 自动进行的); 中间长度的是短破折号 (n dash), 表示数值范围, 如第 33–36 页; 而最长的那个是全破折号 (m dash), 用于复合句。— 通常也就称为破折号。可以分别输入连字符一次、两次或三次来得到这三种不同长度的横线, 即 - 结果为 -, -- 结果为 –, --- 结果为 —。第四种类型的破折号是负号 —, 必须用 \$- 这样的数学模式来输入 (第五章)。

2.6.4 命令字符的显示

在 2.2 节中提到, 字符 # \$ % ^ & \{ } 都被当做命令处理。如果要把它们做为文本直接显示出来, 就必须在它们前面加上字符 \ 或采用其他方法:

```
$ = \$    & = \&    % = \%    # = \#    _ = \_    { = \{
} = \}    \ = \$\backslash$    ^ = \^{}    ~ = \~{}
```

2.6.5 特殊字符 §, †, ‡, ¶, ©, £

在计算机键盘上并没有这些特殊字符, 但可以用如下命令来得到:

```
§ = \S    † = \dag    ‡ = \ddag    ¶ = \P    © = \copyright    £ = \pounds
```

在第五章和附录 A 中描述了如何生成希腊字母和其他数学符号。

2.6.6 外文字母

在非英文的其他欧洲语言中, 还存在一些特殊字符, 它们也可以用 T_EX 得到。这些字符有:

```
œ = {\oe}    Œ = {\OE}    æ = {\ae}    Æ = {\AE}    å = {\aa}    Å = {\AA}
ı = !'        ø = {\o}    Ø = {\O}    l = {\l}    L = {\L}    ß = {\ss}
SS = {\SS}    ¿ = ?'
```

Ångström 可写为 {\AA}ngstr{\o}m, Karlstraße 可用输入 Karlstra{\ss}e 来得到。大写的 \SS 是 L^AT_EX 2_ε 中的新命令。

2.6.7 重音

在欧洲语言中, 有很多发音记号, 也称为重音。在 T_EX 中可以显示出其中绝大多数的重音符号:

```
ô = {\'o}    ó = {\'o}    ô = {\^o}    ö = {\"o}    õ = {\~o}
ō = {\=o}    ô = {\.o}    õ = {\u o}    õ = {\v o}    õ = {\H o}
ō = {\t oo}    õ = {\c o}    õ = {\d o}    õ = {\b o}    õ = {\r o}
```

(最后的命令 \r 是新出现在 L^AT_EX 2_ε 中的。) 上面的 o 只是一个示例, 实际上可以用任何字母。这里应该专门提一下 i 和 j, 在加上重音时必须去掉它们的点, 为此就需要在这两个字母前面加上 \. 命令 \i 和 \j 就会得到 i 和 j。所以 ĩ 和 ĵ 是用 {\u{\i}} 和 {\H{\j}} 来得到的。

对于由非字母组成的重音命令，可以不使用大括号：

`ò=\`o` `ó=\`o` `ô=\`o` `ö=\`o` `õ=\`o` `ō=\`o` `ô=\`o` `ó=\`o`

而由字母组成的重音命令就必须用大括号。在 T_EX 中也可以用其他的记号得到重音，但是这些记号很容易被混淆，因此这里也就不提及那些方法了。

2.6.8 连写

在书籍印刷中，有些特殊组合的字母，并不是单独印出，而是用一个符号来显示，这称为连写。在 T_EX 中对于字母组合 `ff`, `fi`, `fl`, `ffi` 和 `ffl` 不是显示为

`ff`, `fi`, `fl`, `ffi`, `ffl`，而是 `ff`, `fi`, `fl`, `ffi`, `ffl`。

在 3.5.1 节中描述了如何强迫 T_EX 分开显示这些字符，而不是当作连写处理。对于排版类似于 `shelfful` 这样的单词，就必须分开其中的连写，否则排版结果就为 `shelfful`，从单词意义上说这就有点儿不对头。

2.6.9 日期

在文本中的任何地方，都可以用 `\today` 命令显示当前日期。日期显示的标准采用美国的形式，即 月 日，年(如 August 9, 2001)。若要实现其他语言中的日期形式可以借助于 T_EX 命令 `\day`，`\month` 和 `\year`，这三条命令以数字的形式返回当前的日期值。在 10.2.7 节中介绍了显示中文日期的命令。

2.7 脆弱的命令

有些命令的作用不只是局限于它所处的地方，而对文档的其他地方也有影响。例如，类似于 `\chapter{标题}` 这样的章节命令，不但只是在调用它的地方生成标题，而且也有可能在后续页面的顶部以不同的字样显示出来，甚至以第三种字样显示在目录中。类似于这样的可以显示在文档不同地方的参数，称为移动参数。

这样的参数从字面意义上看，是被移动重组的。更精确地说，是未被完全解释的。如果在移动参数中包含了某些命令，那么这些命令会发生变形，从而没有发挥其正常作用。有人形象地称之为分离组合。我们称这样的命令是脆弱的，而其他的不是这样易变的，能抵抗这种重组的命令，则被称为牢固的命令。

从本质上说，所有包含可省参数的命令，如 `\begin` 和 `\end` 都是脆弱的。脆弱命令可以包含在移动参数中，但要在它的前面加上 `\protect` 命令，这样就可以防止其被分离重组。

在移动参数中未用 `\protect` 进行安全保护的脆弱命令，也不是一定要被重组。事实上，只有在极少数情形下才会发生分解。

由于 L^AT_EX 不能正确处理脆弱命令，因此会在屏幕上显示出一长串的错误信息。只要按一下回车键，用户可尝试继续处理下去，而不理睬对该命令是否进行了正确地处理。有时还可能会出现更多的错误，但最终只要按足够多的回车，L^AT_EX 通常是能继续进行后面的处理，除非那个被损坏的命令导致不可能进行下面的处理，而使程序停止运行。

只有下列命令包含移动参数:

- 所有传递文本信息给目录表的命令。它们是章节命令 (3.3.3 节), 如 `\chapter`, `\section` 等, `\addtocontents`, `\addcontentsline` (3.4.3 节) 和 `\caption` (6.7.6 节);
- `\typein` 和 `\typeout` 命令 (8.1.3 节);
- `\markboth` 和 `\markright` 命令 (3.3.1 节);
- 标题页上的 `\thanks` 命令 (3.3.1 节);
- @- 表达式 (6.6.1 节);
- `\bibitem` 的可省参数 (4.3.6 节);
- 调用了 `\makelabels` 命令后的 `\begin{letter}` 命令 (8.10 节)。

只有当脆弱命令出现在上述命令的参数中时, 才有必要用 `\protect` 命令来保护它们。在移动参数中, 对于大多数命令而言, 只要它是脆弱的, 在其前面加上 `\protect`, 对处理没有什么影响。这个规则的例外就是长度命令 (7.2 节) 和计数器命令 (7.1.4 节), 如 `\arabic` 和 `\value`。在这些命令前一定不要加上前缀 `\protect`。

2.8 开始排版第一篇文章

前面简单介绍了 \LaTeX 的基本概念, 对有些复杂情形, 我们只能指出在后面的什么地方可以找到详细内容。初学者没有必要强求很快就理解所有的概念。现在我们假设你要排版一篇文章, 那么下面就具体告诉你该怎么做。注意这里的内容, 在本书的其他部分皆会全面展开, 而且不可避免地存在很多重复。如果你想先完全掌握 \LaTeX 的方法, 然后再使用它, 可以略过本节。

2.8.1 键盘

为此, 我们首先考察一下计算机的键盘。为了输入文本, 我们可以直接利用键盘上的下述按键:

a-z A-Z 0-9 + = * / () [] @

也可以利用下述标点符号:

, ; . ? ! : ' ' - " |

以及空格, Tab 键和回车键。

下面的 10 个特殊字符

\$ % & ~ _ ^ \ { }

是被 \LaTeX 系统移做他用, 如果想在结果中出现上述字符, 需要采用 2.6.4 节给出的方法。除了上述按键以外的其他字符都不允许出现在源文件的普通文本中 (除非经过特殊处理, 例如, 若想输入中文, 必须存成 .ctx 文件, 用 cct 程序转化成 .tex 文件)。不过, 在数学模式 (2.1.3 节) 中, 还可以使用 $\langle \rangle$ | 符号。而在文本模式中, 虽然可以直接利用这三个符号, 但它们分别生成 “i i —” 符号。

2.8.2 输入普通文本

如果想用 \LaTeX 生成纯文本文档, 整个过程是非常简单的, 惟一的困难就在于头文

件的组织。有一种简便方法，那就是照抄已有 L^AT_EX 文件的头文件，或者利用第 8 页上的模板示例。

然后修改 `\title`, `\author` 等命令的参数，在 `\begin{document}` 和 `\end{document}` 之间输入正文。

注意空格的使用。有些特殊符号需要利用 2.6 节中的方法输入，当用到的时候，不妨查一下那一节的内容。但有些符号，在那一节中可能找不到，例如希腊字母 λ 。这时我们就需要浏览附录 A 中的数学符号表，并按下节列出的方法输入。

2.8.3 输入公式

公式的排版是 T_EX 功能最强的部分，可以这样说，当初之所以开发出 T_EX 系统，就是为了解决数学公式的排版这一难题。而 L^AT_EX 进一步完善了 T_EX 数学排版的结构，有利于构造更复杂的公式。另外，我们还可以利用美国数学会给出的 *A_MS-L^AT_EX* 排版更专业的数学文章。我们在第五章中详细讲解了数学公式的排版。

所谓数学环境，就是包围在 `$...$`, `$$...$$`, `\(...\)`, `\[...\]` 等配对符号内部的文本，或者包含在类似于 `\begin{equation}...\end{equation}` 等环境内部的文本，这里 `\(...\)` 等价于 `$...$`，生成与其他文本位于一行的公式，称为正文公式，而 `\[...\]` 与 `$$...$$` 是等价的，生成独占一行的公式，称之为单独公式。`equation` 等环境生成的也是单独公式。

如果想生成数学符号，例如 $y = f(x)$ ，那就必须输入 `$y = f(x)$` 或 `\(y = f(x) \)`。注意数学公式中的空格是没有作用的。而如果想得到

$$\lim_{x \rightarrow a} \frac{f(x) - f(a)}{x - a}$$

那就必须输入

```
$$ \lim_{x \to a} \frac{f(x) - f(a)}{x - a} $$
```

或者

```
\[
\lim_{x \to a} \frac{f(x) - f(a)}{x - a}
\]
```

下面我们看一些常用的数学公式的输入，具体更全面的数学公式的排版，我们留在第五章中详细讲解。以下例子中，一些命令是定义在 `amsmath` 宏包中的，因此需要在文章开始加入指令

```
\usepackage{amssymb,amsmath}
```

- **加减乘除** 一般的加减乘除运算 $a + b$, $a - b$, ab , $a \cdot b$, $a \times b$, a/b , $\frac{a}{b}$ 的输入分别为

```
$a+b$, $a-b$, $a b$, $a \cdot b$, $a \times b$, $a/b$, $\frac{a}{b}$
```

- **上下标** 数学公式中的上标和下标分别用 `^` 和 `_` 表示。例如，要想得到 a_{i1} , a^3 , a^{i_1} ，就需要分别输入

```
$a_{i1}$, $a^3$, $a^{i_1}$
```


- **重音** 我们在 2.6.7 节中已经提到了在正文中如何输入字母重音，这里给出的是数学环境中重音的输入方法，注意两者不能互相代替。

ä	<code>\check{a}</code>	ã	<code>\tilde{a}</code>	á	<code>\acute{a}</code>
à	<code>\grave{a}</code>	â	<code>\dot{a}</code>	â	<code>\ddot{a}</code>
ă	<code>\breve{a}</code>	ā	<code>\bar{a}</code>	→	<code>\vec{a}</code>

- **二项式系数** 可以用 TeX 命令 `\choose` 得到二项式系数，如 `{n \choose m+1}` 结果为 $\binom{n}{m+1}$ 。在 `amsmath` 中定义了 `\binom` 命令，用来输入二项式系数。例如， $\binom{a}{b+c}$ 的输入为 `\binom{a}{b+c}`。如果是在单独公式中，这里的括号是会相应放大的。
- **同余** 在数学中表示同余式的常用输入方法有

$$\begin{aligned}
 a &\equiv b \pmod{p} && \text{\code{\$a \equiv b \pmod{p}}} \\
 a &\equiv b \pmod{p} && \text{\code{\$a \equiv b \pod{p}}} \\
 a \bmod b &= 0 && \text{\code{\$a \bmod b =0}}
 \end{aligned}$$

其中第二个式子需要 `amsmath` 宏包。

- **定界符** 在数学公式中，我们常常需要能自动调整括号等定界符的长度。例如，在排版 $(a+b)^2$ 时，只需输入 `$(a+b)^2$`；而如果排版

$$\left(\frac{a+b}{a+b^2}\right)^2$$

则要输入

```

\[
\left(\frac{a+b}{a+b^2}\right)^2
\]
```

- **函数** 在数学公式中，如果要排版 $\log n$ ，则要输入 `$_\log n$`；而直接输入 `$\log n$`，结果为 $\log n$ ，这是不恰当的。在 A.2.8 节中列出了所有函数的输入方法。
- **省略号** 数学公式中经常出现的三种省略号为： \cdots , \dots , \vdots ，分别通过输入 `$_\cdots$`, `$_\ldots$`, `$_\vdots$` 来得到。第一种省略号经常出现在形如 $1+2+\cdots+n$ 的表达式中；第二种则经常出现在 $i=1,2,\dots,n$ 中；第三种经常出现在矩阵排版中。
- **积分号** 数学公式中，为了排版 $\int_0^1 f(x)dx=2$ ，需要输入 `$_\int_0^1 f(x)dx=2$`。
- **矩阵** `amsmath` 下可以直接使用 `matrix`、`pmatrix` 环境排版矩阵。例如，若输入

```

\[\begin{matrix} 1 & 1+2 & 0.1 \\ 12 & 1 & 0.12 \end{matrix}\]
\[\begin{pmatrix} 1 & 1+2 & 0.1 \\ 12 & 1 & 0.12 \end{pmatrix}\]
```

则可以分别得到

$$\begin{pmatrix} 1 & 1+2 & 0.1 \\ 12 & 1 & 0.12 \end{pmatrix}$$

和

$$\begin{pmatrix} 1 & 1+2 & 0.1 \\ 12 & 1 & 0.12 \end{pmatrix}$$

- **根号** 数学公式中还常用到根式的表示, 要得到 $\sqrt{2}$, $\sqrt[3]{2}$, 则分别输入 `\sqrt{2}`, `\sqrt[3]{2}`.
- **求和、求积** 数学环境中有的符号还可以随着后面的内容一起改变大小, 例如 \sum 和 \prod . 排版

$$\sum_{i=0}^n \frac{1}{i^2} \cdot \prod_{i=1}^n \frac{1}{i^2}$$

的输入为

```
\[
\sum_{i=0}^n \frac{1}{i^2} \cdot \prod_{i=1}^n \frac{1}{i^2}
\]
```

- **文本** 在数学公式中可能还会出现正文文本, 例如:

$$f = \begin{cases} 1, & \text{if } x \geq 0, \\ 0, & \text{otherwise,} \end{cases}$$

我们一般是使用 `\mbox` 命令来输入上述文本的:

```
\[
f=\cases 1, & \mbox{if } x\geq 0, \\ 0, & \mbox{otherwise,} \endcases
\]
```

在排版公式时, 如果公式较长, 那就必须把它分成几行. 这时我们可以使用 `eqnarray*` 环境, 这个环境可以对不同行的公式沿某个位置对齐. 分析下面这个例子, 我们就很容易知道其作用了.

```
\begin{eqnarray*}
G &= & [2x + 2z^6 - z^2 - 3z^4, -10 + z + 3z^3 - 2z^5 + 4y^2, \\
&& 2 - z^3 - 3z^5 + 2z^7]
\end{eqnarray*}
```

$$G = [2x + 2z^6 - z^2 - 3z^4, -10 + z + 3z^3 - 2z^5 + 4y^2, 2 - z^3 - 3z^5 + 2z^7]$$

其中的 `\\` 表示分行. 而两个 `&` 确定了对齐位. 这里在第二行中并没有与第一行 `=` 对齐的字符, 从而可以使等号后面的公式上下对齐.

下面我们循序渐进地看一个比较复杂的例子是怎么实现的:

$$\sum_{i=1}^{\left[\frac{n}{2}\right]} \left(x_{i,i+1}^{\frac{i^2}{2}} \right) \frac{\sqrt{\mu(i)^{\frac{3}{2}}(i^2-1)}}{\sqrt[3]{\rho(i)-2} + \sqrt[3]{\rho(i)-1}}$$

第一步 我们从 $\left[\frac{n}{2}\right]$ 开始, 输入为

```
\left[ \frac{n}{2} \right]
```

第二步 然后输入求和

$$\sum_{i=1}^{\left[\frac{n}{2}\right]}$$

```
\[
  \sum_{i = 1}^{\left[\frac{n}{2}\right]}
\]
```

第三步 接着输入二次项的两个式子:

$$x_{i,i+1}^{i^2} \quad \left[\frac{i+3}{3}\right]$$

```
\[
  x_{i, i + 1}^{i^2} \quad \left[\frac{i + 3}{3}\right]
\]
```

第四步 然后排版整个二次项式子:

$$\left(x_{i,i+1}^{i^2}\right)^{\left[\frac{i+3}{3}\right]}$$

```
\[
  \binom{x_{i,i + 1}^{i^2}}{\left[\frac{i + 3}{3}\right]}
\]
```

第五步 排版分子中的根式 $\sqrt{\mu(i)^{\frac{3}{2}}(i^2-1)}$:

```
\sqrt{\mu(i)^{\frac{3}{2}}(i^2-1)}
```

第六步 然后排版分母的三次根式 $\sqrt[3]{\rho(i)-2}$ 和 $\sqrt[3]{\rho(i)-1}$:

```
\sqrt[3]{\rho(i)-2} \quad \sqrt[3]{\rho(i)-1}
```

第七步 之后我们得到分式

$$\frac{\sqrt{\mu(i)^{\frac{3}{2}}(i^2-1)}}{\sqrt[3]{\rho(i)-2} + \sqrt[3]{\rho(i)-1}}$$

```
\[
  \frac{\sqrt{\mu(i)^{\frac{3}{2}}(i^2-1)}}{\sqrt[3]{\rho(i)-2} + \sqrt[3]{\rho(i)-1}}
\]
```

第八步 最后我们得到整个式子的排版:

$$\sum_{i=1}^{\left[\frac{n}{2}\right]} \left(x_{i,i+1}^{i^2}\right)^{\left[\frac{i+3}{3}\right]} \frac{\sqrt{\mu(i)^{\frac{3}{2}}(i^2-1)}}{\sqrt[3]{\rho(i)-2} + \sqrt[3]{\rho(i)-1}}$$

```
\[
  \sum_{i = 1}^{\left[\frac{n}{2}\right]}
\]
```

```

\binom{ x_{i, i + 1}^{i^2} }
{ \left[ \frac{i + 3}{3} \right] }
\frac{ \sqrt{ \mu(i)^{\frac{3}{2}} (i^2 - 1) } }
{ \sqrt[3]{\rho(i) - 2} + \sqrt[3]{\rho(i) - 1} }
\]

```

2.8.4 公式示例

下面给出一组公式排版示例，有些公式中的一些命令上面可能没介绍到，但是比较排版结果和输入格式，可以从中领悟出一些用法，面进一步的细节，可以参看本书的第五章。

公式 1

$$x \mapsto \{c \in C \mid c \leqslant x\}$$

```

\[
x \mapsto \{\, , c \in C \mid c \leqslant x \, \}
\]

```

`\leqslant` 为 `amssymb` 中的命令。

公式 2

$$\left| \bigcup (I_j \mid j \in J) \right| < m$$

```

\[
\left| \bigcup (\, , I_{\{j\}} \mid j \in J \, ,) \right|
< \mathfrak{m}
\]

```

公式 3

$$A = \{x \in X \mid x \in X_i \text{ for some } i \in I\}$$

```

\[
A = \{\, , x \in X \mid x \in X_{\{i\}}
\quad \text{for some } i \in I \, \}
\]

```

公式 4

$$\langle a_1, a_2 \rangle \leq \langle a'_1, a'_2 \rangle \quad \text{iff} \quad a_1 < a'_1 \quad \text{or} \quad a_1 = a'_1 \text{ and } a_2 \leq a'_2$$

```

\[
\langle a_{\{1\}}, a_{\{2\}} \rangle \leq \langle a'_{\{1\}}, a'_{\{2\}} \rangle
\quad \text{iff} \quad a_{\{1\}} < a'_{\{1\}} \quad \text{or} \quad
a_{\{1\}} = a'_{\{1\}} \quad \text{and} \quad a_{\{2\}} \leq a'_{\{2\}}
\]

```

公式 5

$$\Gamma_{u'} = \{\gamma \mid \gamma < 2\chi, B_\alpha \not\subseteq u', B_\gamma \subseteq u'\}$$

```
\[
\Gamma_{u'} = \{\, \gamma \mid \gamma < 2\chi,
\ B_{\alpha} \not\subseteq u', \ B_{\gamma} \subseteq u' \,\}
\]
```

公式 6

$$A = B^2 \times \mathbb{Z}$$

```
\[
A = B^{2} \times \mathbb{Z}
\]
```

公式 7

$$\left(\bigvee(s_i \mid i \in I)\right)^c = \bigwedge(s_i^c \mid i \in I)$$

```
\[
\left(\bigvee(\, s_i \mid i \in I \,)\right)^c =
\bigwedge(\, s_i^c \mid i \in I \,)\]
\]
```

公式 8

$$y \vee \bigvee([B_\gamma] \mid \gamma \in \Gamma) \equiv z \vee \bigvee([B_\gamma] \mid \gamma \in \Gamma) \pmod{\Phi^x}$$

```
\[
y \vee \bigvee(\, [B_{\gamma}] \mid \gamma
\in \Gamma \,)\equiv z \vee \bigvee(\, [B_{\gamma}]
\mid \gamma \in \Gamma \,)\pmod{\Phi^x}
\]
```

公式 9

$$f(\mathbf{x}) = \bigvee_m \left(\bigwedge_m (x_j \mid j \in I_i) \mid i < \aleph_\alpha \right)$$

```
\[
f(\mathbf{x}) = \bigvee\nolimits_{\mathfrak{m}}
\left(\, \bigwedge\nolimits_{\mathfrak{m}}(x_j \mid j \in I_i) \mid i < \aleph_{\alpha}
\right)
\]
```

公式 10

$$F(x)|_a^b = F(b) - F(a)$$

```
\[
\left. F(x) \right|_{\{a\}^{\{b\}}} = F(b) - F(a)
\]
```

公式 11

$$u + v \underset{\alpha}{\sim} w \overset{2}{\sim} z$$

```
\[
u \underset{\{\alpha\}}{+} v \overset{1}{\thicksim} w
\overset{2}{\thicksim} z
\]
```

公式 12

$$f(x) \stackrel{\text{def}}{=} x^2 - 1$$

```
\[
f(x) \overset{\text{\text{def}}}{=} x^{\{2\}} - 1
\]
```

公式 13

$$\widehat{a + b + \cdots + z}^n$$

```
\[
\overbrace{a + b + \cdots + z}^{\{n\}}
\]
```

公式 14

$$\left| \begin{array}{cc} a + b + c & uv \\ a + b & c + d \end{array} \right| = 7$$

$$\left\| \begin{array}{cc} a + b + c & uv \\ a + b & c + d \end{array} \right\| = 7$$

```
\[
\begin{vmatrix}
a + b + c & uv \\
a + b & c + d
\end{vmatrix}
\]
```

```

= 7
\]

\[
\begin{Vmatrix}
a + b + c & uv \\
a + b & c + d
\end{Vmatrix}
= 7
\]

```

公式 15

$$\sum_{j \in N} b_{ij} \hat{y}_j = \sum_{j \in N} b_{ij}^{(\lambda)} \hat{y}_j + (b_{ii} - \lambda_i) \hat{y}_i \hat{y}$$

```

\[
\sum_{j \in \mathbf{N}} b_{ij} \hat{y}_j =
\sum_{j \in \mathbf{N}} b^{(\lambda)}_{ij} \hat{y}_j +
(b_{ii} - \lambda_i) \hat{y}_i \hat{y}
\]

```

公式 16

$$\left(\prod_{j=1}^n \hat{x}_j \right) H_c = \frac{1}{2} \hat{k}_{ij} \det \hat{\mathbf{K}}(i|i)$$

$$\left(\prod_{j=1}^n \hat{x}_j \right) H_c = \frac{1}{2} \hat{k}_{ij} \det \hat{\mathbf{K}}(i|i)$$

```

\[
\left( \prod_{j=1}^n \hat{x}_j \right) H_c =
\frac{1}{2} \hat{k}_{ij} \det \hat{\mathbf{K}}(i|i)
\]

```

```

\[
\biggl( \prod_{j=1}^n \hat{x}_j \biggr) H_c =
\frac{1}{2} \hat{k}_{ij} \det \widehat{\mathbf{K}}(i|i)
\]

```

公式 17

$$\det \mathbf{K}(t=1, t_1, \dots, t_n) = \sum_{I \in \mathbf{n}} (-1)^{|I|} \prod_{i \in I} t_i \prod_{j \in I} (D_j + \lambda_j t_j) \det \mathbf{A}^{(\lambda)}(I|I) = 0$$

```
\[
\det \mathbf{K} (t = 1, t_{\{1\}}, \dots, t_{\{n\}}) =
\sum_{I \in \mathbf{n}} (-1)^{|I|}
\prod_{i \in I} t_{\{i\}}
\prod_{j \in I} (D_{\{j\}} + \lambda_{\{j\}} t_{\{j\}})
\det \mathbf{A}^{(\lambda)} (\overline{I} | \overline{I}) = 0
```

```
\]
```

公式 18

$$\lim_{(v,v') \rightarrow (0,0)} \frac{H(z+v) - H(z+v') - BH(z)(v-v')}{\|v-v'\|} = 0$$

```
\[
```

```
\lim_{(v, v') \rightarrow (0, 0)}
\frac{H(z + v) - H(z + v') - BH(z)(v - v')}{\|v - v'\|} = 0
```

```
\]
```

公式 19

$$\int_{\mathcal{D}} |\partial u|^2 \Phi_0(z) e^{\alpha|z|^2} \geq c_4 \alpha \int_{\mathcal{D}} |u|^2 \Phi_0 e^{\alpha|z|^2} + c_5 \delta^{-2} \int_A |u|^2 \Phi_0 e^{\alpha|z|^2}$$

```
\[
```

```
\int_{\mathcal{D}} |\overline{\partial} u|^2
\Phi_0(z) e^{-\alpha|z|^2} \geq
c_4 \alpha \int_{\mathcal{D}} |u|^2 \Phi_0
e^{-\alpha|z|^2} + c_5 \delta^{-2} \int_A
|u|^2 \Phi_0 e^{-\alpha|z|^2}
```

```
\]
```

公式 20

$$A = \begin{pmatrix} \varphi \cdot X_{n,1} & (x + \varepsilon_2)^2 & \cdots & (x + \varepsilon_{n-1})^{n-1} & (x + \varepsilon_n)^n \\ \varphi_1 \times \varepsilon_1 & \varphi \cdot X_{n,2} & \cdots & (x + \varepsilon_{n-1})^{n-1} & (x + \varepsilon_n)^n \\ \varphi_2 \times \varepsilon_1 & \varphi_2 \times \varepsilon_2 & \cdots & (x + \varepsilon_{n-1})^{n-1} & (x + \varepsilon_n)^n \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \varphi \cdot X_{n,1} & \varphi \cdot X_{n,2} & \cdots & \varphi \cdot X_{n,n-1} & \varphi \cdot X_{n,n} \\ \varphi_n \times \varepsilon_1 & \varphi_n \times \varepsilon_2 & \cdots & \varphi_n \times \varepsilon_{n-1} & \varphi_n \times \varepsilon_n \end{pmatrix} + I_n$$

```
\[
```

```
\mathbf{A} =
\begin{pmatrix}
\frac{\varphi \cdot X_{n,1}}{\varphi_1 \times \varepsilon_1} & \cdots & \frac{\varphi \cdot X_{n,n}}{\varphi_n \times \varepsilon_n} \\
\vdots & \ddots & \vdots \\
\frac{\varphi \cdot X_{n,1}}{\varphi_n \times \varepsilon_1} & \cdots & \frac{\varphi \cdot X_{n,n}}{\varphi_n \times \varepsilon_n}
\end{pmatrix} + I_n
```



```

& (x + \varepsilon_{n - 1})^{n - 1}
& (x + \varepsilon_n)^n \\
\dfrac{\varphi \cdot X_{n, 1}}{\{\varphi_{\{2\}} \times \varepsilon_{\{1\}}\}}
& \dfrac{\varphi \cdot X_{n, 2}}{\{\varphi_{\{2\}} \times \varepsilon_{\{2\}}\}}
& \cdots & (x + \varepsilon_{n - 1})^{n - 1}
& (x + \varepsilon_n)^n \\
\hdotsfor{5} \\
\dfrac{\varphi \cdot X_{n, 1}}{\{\varphi_n \times \varepsilon_{\{1\}}\}}
& \dfrac{\varphi \cdot X_{n, 2}}{\{\varphi_n \times \varepsilon_{\{2\}}\}}
& \cdots & \dfrac{\varphi \cdot X_{n, n - 1}}{\{\varphi_n \times \varepsilon_{\{n - 1\}}\}}
& \dfrac{\varphi \cdot X_{n, n}}{\{\varphi_n \times \varepsilon_{\{n\}}\}}
\end{pmatrix}
+ \mathbf{I}_n
\]

```

公式 21 (需要 amscd 宏包)

```

A \xrightarrow{\log} B \xrightarrow[\text{bottom}]{} C \xlongequal{\quad} D \xleftarrow{\quad} E \xleftarrow{\quad} F
\downarrow \text{one-one} \quad \downarrow \quad \uparrow \text{onto}
X \xlongequal{\quad} Y \longrightarrow Z
\uparrow \beta \quad \uparrow \gamma
D \xrightarrow{\alpha} E
\]
\begin{CD}
A @>\log>> B @>>\text{bottom}> C @= \\
@. @<<< E @<<< F \\
@V\text{one-one}VV @VVV @AA\text{onto}A \\
X @= Y @>>> Z \\
@A\beta AA @AA\gamma A \\
D @>\alpha>> E
\end{CD}
\]

```

注：如果想输入键盘上没有的符号运算符，可以首先找一下附录 A，找到后，输入它的名称，但是必须把它包围在数学环境内部。

2.8.5 生成结果

在一般情形中，我们刚开始处理的文档都不会很长，因此一般不需要太复杂的结构。如果读者想了解关于 L^AT_EX 文本处理的更多细节，以及如何排版更复杂的公式，请仔细阅读本书的第二部分。

当输入完一篇简单的文章后，我们可以按 2.1.4 节中的方法编译处理。如果遇到了错误，可以查阅附录 D，从中得到帮助。实际上，第一次用 L^AT_EX 处理文档时，一般都会遇到错误。最简单的应付方法就是直接按〈回车〉。然后浏览结果，从而有可能知道自己错在何处。

第二部分 高级 L^AT_EX

第三章 页面组织

3.1 文档类

在 L^AT_EX 源文件导言中的第一条命令通常用来确定整篇文档的全局处理格式。该条命令的语法是:

```
\documentclass[选项]{类}    或  
\documentstyle[选项]{类}
```

其中后者只适用于 L^AT_EX 2.09。对于 L^AT_EX 2_ε, 可以用 `\documentstyle` 命令, 但这只是为了与原来的源文件兼容而提供的。

而类可以取的值是: `book`, `report`, `article`, `letter`, `cctart` 或 `cctbook`, 但每次只能选取一种做为类的值。其中最后两个类用于 CCT 中文 L^AT_EX 2_ε, 详情见第四部分。这些类之间的差异包含页面布局和组织方式的不同。一篇 `article`(论文) 可以有 `parts`(部分), `sections`(节), `subsections`(小节), 等等, 而一篇 `report`(报告) 还可以有 `chapters`(章)。一本 `book`(书) 也有 `chapters`(章), 同时对奇偶页要区别对待; 而且它根据章节标题在每页上打印出当前的页眉。

3.1.1 标准的类选项

用选项可以对格式进行各种不同的修改。它们可如下分组:

选择字体尺寸

可以用如下选项来选择基本的字体尺寸:

```
10pt  11pt  12pt
```

这个选项就是在文档中普通文本所取的字体尺寸。默认值是 10pt, 这意味着如果没有指定尺寸选项时就用这个值。其他所有字体尺寸都是相对于这个标准尺寸而言的, 因此如果选定了不同的基本字体尺寸, 节的标题, 页脚等等就会相应地自动改变尺寸。(注意在 L^AT_EX 2.09 中没有 10pt 选项; 如果想用 10pt 作为基本尺寸, 只要不指定尺寸选项就可以了。)

指定纸张大小

L^AT_EX 是根据选定的字体尺寸和纸张大小来计算行宽和每页行数。同时它也会选择适当的页边, 以使文本水平和竖直居中。为了做到这一点, 就需要知道所用纸张的格式。

下面的选项可以用来指定纸张大小, 所有选项都只能用于 L^AT_EX 2_ε:

<code>letterpaper(11 × 8.5 in)</code>	<code>a4paper(29.7 × 21 cm)</code>
<code>legalpaper(14 × 8.5in)</code>	<code>a5paper(21 × 14.8 cm)</code>
<code>executivepaper(10.5 × 7.25 in)</code>	<code>b5paper(25 × 17.6 cm)</code>

默认值是 `letterpaper`, 即纸张大小采用美国信纸的尺寸 (11 × 8.5 in)。

通常情况下, 纸张格式中长的方向是竖直方向, 即所谓的纵向模式。利用选项

`landscape`

可以使短的方向成为竖直方向, 即横向模式。

页面格式

在一页上的文本可以用下面的选项来构成单列或双列版式:

`onecolumn` `twocolumn`

默认值是 `onecolumn`。当采用 `twocolumn` 选项时, 两列间距以及列间可能存在的竖线的宽度用 `\columnsep` 和 `\columnseprule` 来指定, 后面将会讲到它们。

要想奇偶页的页码打印方式不一样, 可以用选项

`oneside` `twoside`

来指定。当用的是 `oneside` 时, 所有页码的打印方式是一样的; 然而, 当用的是 `twoside` 时, 在页眉上的活动标题中, 若当前页码为奇数, 页码出现在右边, 若为偶数, 页码出现在左边。注意它并不是强迫打印机双面打印。这里的想法是当以后真的双面印刷时, 页码总是在每页的外侧, 阅读时容易看到。这是 `book` 类的默认值。对于 `article` 和 `report` 类, 默认值是 `oneside`。

对于 `book` 类, 每一章通常都是开始于右边, 即从奇数页开始。选项

`openright` `openany`

可以控制这个功能: 用 `openany` 时, 总是在下一页上开始新的一章, 而如果选择了默认值 `openright`, 必要时可以插上一空页, 以保证新的一章在奇数页上开始。

通常 `book` 或 `report` 的标题是单独一页的, 而在 `article` 中, 标题则是与开头的文本在同一页上。利用选项

`notitlepage` `titlepage`

可以重定义这种标准行为。请参见 3.3.1 节和 3.3.2 节。

其他选项

还有如下一些标准选项:

`leqno` 单独公式中的公式编号出现在左边, 而不是像通常那样放在右边 (5.1 节)。

`fleqn` 单独公式左对齐, 而不是居中 (5.1 节)。可以用参数 `\mathindent` (下面会讲到它) 来设置缩进大小。

openbib 参考文献的格式可以改变成每一片断位于一个新行上。默认方式下，每个条目的文本都是聚在一起的。

draft 如果 L^AT_EX 的断行机制不能很好的发挥作用，总有些文本在右边界突出，那么这个选项就会用一个粗黑条来标识它，从而很容易发现突出的行。

final 与 **draft** 相反，这是默认值。无论文本行有多宽，也不会加上标识。

如果同时给出了多个选项，那就要用逗号把它们分开，例如，

```
\documentclass[11pt,twoside,fleqn]{article}
```

选项顺序是无关紧要的。如果同时指定了两个矛盾的选项，比如说，同时用了 **oneside** 和 **twoside**，无法确定哪个将发挥作用。这主要看在类文件中的具体定义了，因此最好避免出现这种情形。

与某些选项相关的参数

有些选项要使用一些已有确定默认值的参数：

\mathindent

当选择了 **fleqn**(5.1 节) 时，指定公式相对于左边界的缩进量；

\columnsep

为 **twocolumn** 选项指定两列间距；

\columnseprule

为 **twocolumn** 选项确定两列间竖线的宽度。默认值是宽度为零，即没有竖线。

这些参数的标准值可以用 L^AT_EX 命令 **\setlength** 来修改。例如，要把 **\mathindent** 改为 2.5cm，可以用

```
\setlength{\mathindent}{2.5cm}
```

对这些参数的修改，既可以在导言中进行，也可以在文档的其他任意地方进行。在导言中进行的修改，适用于整个文档，而在正文内的修改，其作用终止于下一次修改，或者它所在的环境结束(2.4 节)。在后一种情形中，先前的值就开始继续发挥作用。

3.1.2 利用宏包增加功能

虽然文档类确定了文档所具有的一般性质，如页面布局和章节划分，但是也可以用一些更具体的宏包，来改变某些命令的行为，甚至定义一些全新的命令，以增加非 L^AT_EX 标准的其他功能。在 L^AT_EX 的发行中，有很多这样的宏包，甚至你也可以从 L^AT_EX 用户组织那里获取成千个可用的宏包(8.8.6 节)，当然你也可以自己编写(附录 C)。

一个宏包就是一组 L^AT_EX(或 T_EX) 命令集，并且被保存在一个后缀为 **.sty** 的文件中，有些特殊命令只能用在宏包中。要调用一个宏包，只需在导言中用

```
\usepackage{宏包}
```

这里的宏包就是文件的基本名。用一条 **\usepackage** 可以调用多个宏包。例如，在标准 L^AT_EX 中有两个宏包，保存在文件 **makeidx.sty**(8.4 节) 和 **ifthen.sty**(7.3.5 节) 中，那么可以用如下方式调用它们：

```
\usepackage{makeidx,ifthen}
```

一个宏包也可以具有与之相关的选项，其选择方式与文档类选项的选择方式相同，即要把选项名放在中括号内。其一般的语法是：

```
\usepackage[选项 1, 选项 2 ...]{宏包 1, 宏包 2, ... }
```

这里所列出的选项将适用于所有选择的宏包。如果有的宏包不理解所给选项，将在显示器上给出一条警告信息。

至于有哪些选项可用，那就要看具体的宏包了。你必须去阅读随宏包一起发行的手册或文档，不但要看看到底有哪些选项，还要看看这个宏包是做什么的，它定义了那些新命令。

3.1.3 样式选项

在 L^AT_EX 2.09 中并没有 `\usepackage` 命令；事实上，若在用 `\documentclass` 的地方换上 `\documentstyle`，`\usepackage` 命令就根本没有作用。在 L^AT_EX 2.09 中，宏包与样式选项的处理方法相同，也就是要把宏包文件的基本名加到 `\documentstyle` 命令的选项表中。正是由于这个原因，我们也称它们为样式选项文件，或简称为样式文件(Style File)，这也就是这些文件后缀 `.sty` 的来源。在 L^AT_EX 2_ε 中继续使用这个后缀，是为了保证原来的样式文件还可以作为宏包使用。

在真正的选项(在类或宏包文件中对其进行程序设计)和宏包(从与之同名的文件中读取)之间是有着本质区别的。而在 L^AT_EX 2.09 中，这种差别是不明确的，经常导致混淆。

在原来的 L^AT_EX 版本中，要想向 `article` 类(或“主样式”)中调入宏包 `makeidx`，同时带有选项 `12pt` 和 `titlepage`，就必须用

```
\documentstyle[makeidx,12pt,titlepage]{article}
```

由于这里是把宏包做为选项来处理，所以它们自己不可能再有选项了。

3.1.4 全局和局部选项

用 `\documentclass` 命令指定的选项有一个有趣的特点，那就是它适用于所有后面的宏包。这就是说，如果有几个宏包具有相同的选项，那就只需在 `\documentclass` 中声明一次。例如，你可以设计一个宏包，修改 `article`，以生成一个特殊的样式，使得当文本为单列或双列时做不同的事情；这个宏包可以用类选项 `onecolumn` 和 `twocolumn` 来实现这一点。或者对 `draft` 选项进行精细加工，以生成类似于手稿那样的双倍行距。与之相应的，几个宏包可能具有与语言有关的特征，用 `french` 或 `german` 选项来激活；那么只要在 `\documentclass` 命令中声明就可以适用于所有的宏包。这样的选项称为全局的，因为它对后续的所有宏包都有作用。

全局选项并不一定只限于列在 3.1.1 节中的那些标准类选项。如果类和任一宏包都不理解所列出的一个或多个选项，只会显示出一条警告信息。相反地，由 `\usepackage` 命令指定的选项只适用于列于该条命令中的宏包；而且它应适用于其中每一宏包。如果那些宏包中的一个或多个不认识任一个局部选项，都会显示出一条警告信息。

3.2 页面样式

基本的页面格式是由样式参数决定的。除了一种特殊情形外，该命令通常放在导言中。其形式为：

`\pagestyle{ 样式 }`

这里的不可省参数 样式 可以取如下的值：

plain 页面的页眉是空的，页脚由居中的页码组成。当在导言中没有 `\pagestyle` 时，这是默认值。

empty 页眉和页脚都是空的；也不显示页码。

headings

页眉由页码及文档类所决定的标题信息（章节标题）组成；页脚为空。对每一章的第一页没有作用。

myheadings

同 **headings** 差不多，只是页眉的标题不是自动选取，而是由 `\markright` 或 `\markboth` 命令显式决定（见下面的解释）。

命令

`\thispagestyle{ 样式 }`

的作用同 `\pagestyle` 一样，但它只对当前页起作用。例如，如果想取消当前页的页码，可以利用命令 `\thispagestyle{empty}`。这实际上只是不显示页码，下一页的页码就与没有用该命令一样。

3.2.1 生成页眉的声明

对于页面样式 **headings** 和 **myheadings**，出现在页眉中的信息可以用下面的声明来确定：

`\markright{ 右边纸页眉 }`

`\markboth{ 左边纸页眉 }{ 右边纸页眉 }`

`\markboth` 声明适用于 `\twoside` 文档类选项，这时假定偶数页位于左边，奇数页位于右边。而且，对左边页，页码显示在页眉的左边，对右边页，页码显示在页眉的右边。

对于单面输出，认为所有页都是右手方向的。在这种情况下，应该用声明 `\markright`。但对双面输出，也可以用它来重新定义 `\markboth` 中的右边纸页眉。

对于 **headings** 页面样式，位于页眉上的标准信息是章、节和小节的标题，具体要看文档和页面样式，对此有如下的图解：

样式		左边纸	右边纸
book, report	单面页	—	章
	双面页	章	节
article	单面页	—	节
	双面页	节	小节

如果在一页上有不只一个 `\section` 或 `\subsection`, 就在页眉上显示最后那一个的内容。

3.2.2 生成复杂页眉与页脚

前面给出的 `\markright` 或 `\markboth` 命令可以对页眉进行所需的改变。如果在这两条命令的参数中输入文本, 那么所输入的文本就会显示在页眉中。但是通常我们希望页眉具有更复杂的样式, 随便翻开一本精心印刷的书籍, 我们会发现它的页面样式一般为在奇数页的页眉居中显示节标题, 右边显示页码; 偶数页的页眉居中显示章标题, 右边显示页码。在页眉与正文之间有一条横线。页脚则是空的 (每章首页的页眉是空的, 只在页脚中显示页码)。本书的页面样式就是这样的。为了做到这一点, 我们需要利用命令 `\leftmark` 与 `\rightmark`。下述命令定义就可以得到这样的页面样式:

```
\makeatletter
\def\@evenhead{\pushziti
\vbox{\hbox to\textwidth{\rlap{第 \textbf{\thepage} 页 }
\hfil{\leftmark}\llap{}\hfil\mbox{}}}
\protect\vspace{2truemm}\relax
\hrule depth0pt height0.15truemm width\textwidth
}\popziti}
\def\@oddhead{\pushziti
\vbox{\hbox to\textwidth{\mbox{}\hfil \rlap{}{\rightmark}}
\hfil\llap{第 \textrm{\thepage} 页 }}
\protect\vspace{2truemm}\relax
\hrule depth0pt height0.15truemm width\textwidth
}\popziti}
\def\@evenfoot{}
\def\@oddfoot{}
\makeatother
```

注意这里采用了一些 CCT emTeX 中的命令 (如 `\pushziti`, `\popziti`), 如果是在英文 emTeX 中设计这样的页面样式, 则需要去掉这些命令以及其中的汉字, 这对结果没有影响。

3.2.3 页的编号

定义页编号的声明形式如下:

```
\pagenumbering{数字形式}
```

数字形式可取如下值:

arabic 通常 (阿拉伯) 数字,
roman 小写罗马数字,
Roman 大写罗马数字,
alpha 小写字母
Alpha 大写字母。

标准值是 arabic。这个声明把当前页码重新设置为 1。为了用罗马数字显示前言的页码,而正文部分的页码用阿拉伯数字显示,而且第一章开始的页码为 1,那就必须在开始前言时加上 `\pagenumbering{roman}` 声明,然后紧接在第一个 `\chapter` 命令后面,利用 `\pagenumbering{arabic}` 对其进行重设。(在 3.3.5 节中有另一种方法。)

如果想使页码不是从 1 开始编号,那么可以使用命令

`\setcounter{page}{页码}`

这里的 页码 是前一页的编号。

3.2.4 与段落有关的距离

下述参数对段落的形式有影响,可以用在 7.2 节介绍的 `\setlength` 命令来给它们设定新值:

`\parskip`

两个段落之间的距离。以 ex 为单位,可以使它随着字体尺寸而自动改变。其应是一个弹性长度。

`\parindent`

段落中第一行的缩进量。

`\baselinestretch`

这是一个度量两条基线之间正常间距的数值,所谓基线就是字母所坐的位置,即忽略了类似于 g 和 y 等字母的下挂部分后的位置。这个值初始化为 1,表示标准的线距。它可以用下面的命令改值:

`\renewcommand{\baselinestretch}{因子}`

这里的 因子 是十进制浮点数,如 1.5 表示增加了 50%。它适用于所有的字体尺寸。如果这条命令没有放在导言中,那么要直到选择了另一条字体尺寸命令,其才会发挥作用 (4.1.2 节)。

这些参数既可以放在导言中,也可以放在文档的其他任何地方。在后一种情况里,这种修改的作用持续到下一次修改为止,或者它所在的环境结束 (2.4 节)。

真正的行间距是包含在 `\baselineskip` 这个长度参数中,只要声明了一个字体尺寸,就会自动设置其值。对每一种字体尺寸,都存在一个标准值,把它乘上 `\baselinestretch`,就得到 `\baselineskip`。详情请见 4.1.2 节。

如果在一个段落中间修改 `\baselineskip`,新值就确定了整段的行间距。更精确地说,在一段结束时的值才有效。

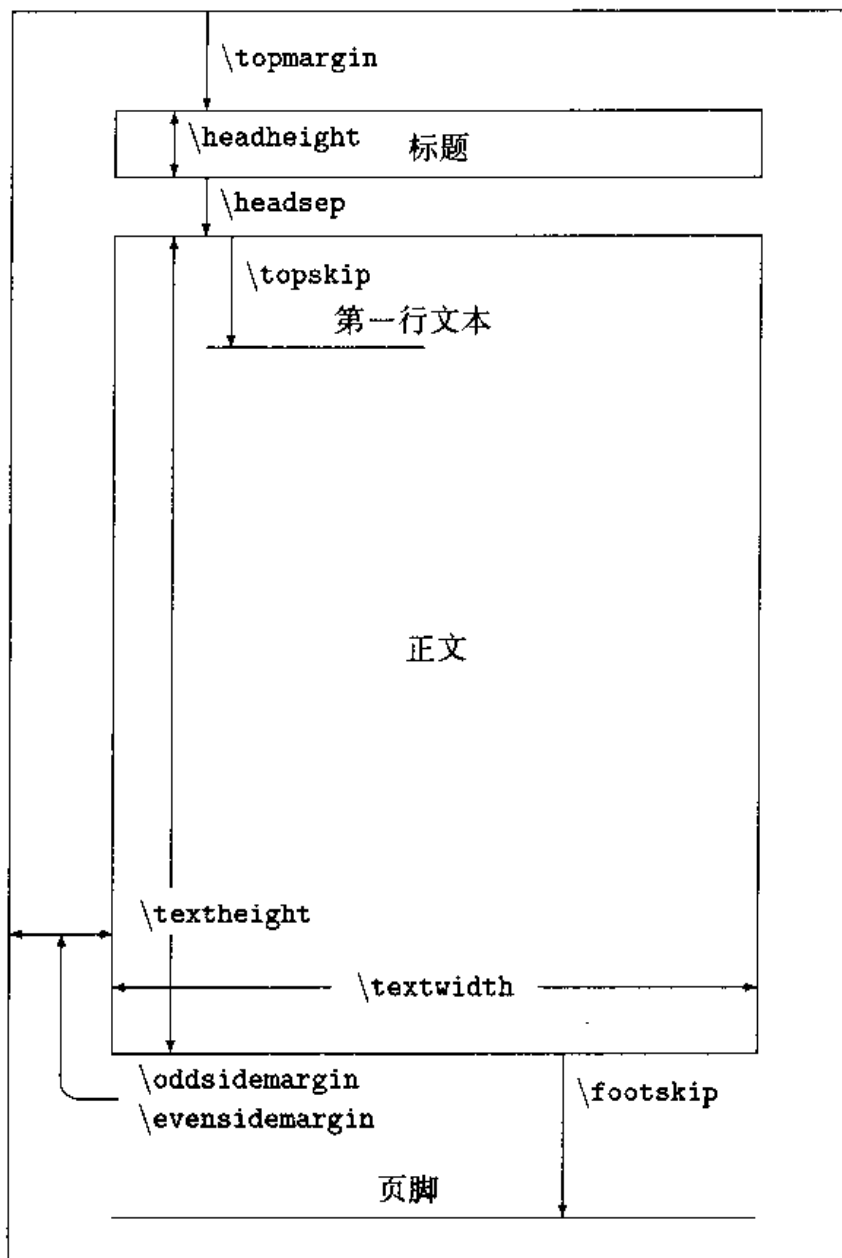


图 3.1 页面布局参数

3.2.5 页面格式

每一页都是由页眉、包含实际文本的正文，以及页脚组成的。所谓页面格式的选择就是确定在页眉和页脚中应包含哪些信息。

对于页眉，正文与页脚之间的距离，上页边界和左页边界，文本行宽，以及页眉、正文和页脚的高度， \LaTeX 都有默认值。在图 3.1 中标示这些长度，它们的意义如下：

- `\oddsidemargin` 奇数页的左边界，
- `\evensidemargin` 偶数页的左边界，
- `\topmargin` 从上页边到页眉顶的距离，
- `\headheight` 页眉的高度，

`\headsep` 页眉基线到正文顶部的距离,
`\topskip` 从正文顶部到正文第一行的基线之间的距离,
`\textheight`, `\textwidth` 主要正文的高度和宽度,
`\footskip` 表示从正文底部到页脚底部的距离,
`\paperwidth`, `\paperheight` 由纸张大小选项所确定的总宽度和高度, 包含所有的页边,
`\footheight` 已过时, 在 \LaTeX 2_ϵ 中已被去掉。

可以利用命令 `\setlength`(7.2 节) 给这些长度指定一个新值, 这些修改最好放在导言中。例如, 可以用

```
\setlength{\textwidth}{12.5cm}
```

使文本行宽变为 12.5cm。

在 \LaTeX 2.09 中还定义了一个参数 `\footheight`, 但从来没有用到它。因此在 \LaTeX 2_ϵ 中就去掉了这个参数。

为了能准确计算页面布局, 我们必须知道 \LaTeX 中的度量是从纸张上边一英寸的点和纸张左边一英寸的点开始的。因此整个左边界是 `\oddsidemargin` 加上一英寸。 \LaTeX 2_ϵ 的页面参数 `\paperwidth` 和 `\paperheight` 中已经包含了这额外的一英寸, 它们是通过 `\documentclass` 中的纸张大小选项赋值的, 在内部用它来计算页边界, 从而使文本居中。用户也可以使用这两个参数。

对于文档类 `book` 或者选项 `twoside`, 每一页上正文的底边恰好处在相同的位置上。而对于其他的类或选项, 这个位置就可能稍有点儿变化。在前面那两种情形中, 常量底边是用内部命令 `\flushbottom` 来得到的, 而变化的底边是用命令 `\raggedbottom` 生成的。用户也可以用这两个声明来随时改变底边的形式, 从而使之与文档类和选项无关。

3.2.6 单双列页面

文档类选项 `twocolumn` 可以使整篇文档的每页都是双列版式显示。而默认值是每页为单列形式。如果要使单独某一页以单列或双列形式排版, 可以用下面的声明来达此目的:

```
\twocolumn[ 文本 ]
```

它中止当前页, 用两列形式排版后面的每一页。这里的可省略参数 `文本` 要用单列形式显示在页面顶部, 宽度与页宽相同。

```
\onecolumn
```

中止当前的双列页面形式, 用单列形式排版后续的每一页。

选项 `twocolumn` 会自动地改变某些相对于单列格式时的样式参数, 例如缩进量。而使用 `\twocolumn` 命令就不会有这种好处。如果你想要进行这种改变, 那就须用相应的 `\setlength` 声明来实现这一点。如果整篇文档都是双列格式, 那么使用类选项不失为一种最好的方案。

还有一个页面样式参数 `\columnwidth`, 它表示一列文本的宽度。对于单列文本, 它与 `\textwidth` 是相同的, 当选择的是 `twocolumn` 时, \LaTeX 会根据 `\textwidth` 和

`\columnsep` 的值来计算它的值。文档作者不应该改变这个值，但可以利用它，例如画一个与列文本同宽的标尺。

3.3 文档中的部分

每一篇文档都要被分成章、节、小节等等。在结尾处还可能有附录，开头处有标题页，目录和摘要等等。在 \LaTeX 中有许多可用的命令，把用户从这种复杂的格式考虑中解脱出来。除此而外，还可以进行连续的标题编号和子编号。甚至目录也可以自动生成和显示出来。

某些章节命令的作用与所选用的文档类有关，并不是在所有的类中都可以用所有的命令。

3.3.1 标题页

可以用如下环境来生成没有格式的标题页：

```
\begin{titlepage} 标题页文本 \end{titlepage}
```

或者利用如下命令，从而采用 \LaTeX 预定义好的标题页格式：

```
\title{ 标题文本 }
```

```
\author{ 作者名与地址 }
```

```
\date{ 日期文本 }
```

在标准的 \LaTeX 标题页格式中，所有内容将来都是以居中的形式出现的。如果标题太长，也会自动断行。作者自己可以用 `\\` 命令来选择断行点，也就是说所用形式为

```
\title{...\\...\\...}.
```

如果同时要列出几个作者，那么可以用 `\and` 分开姓名，如 `\author{G. Smith \and J. Jones}`，这些姓名将来会打印在一行上。而

```
\author{ 作者 1\\ 学术团体 1\\ 地址 1
```

```
\and 作者 2\\ 学术团体 2\\ 地址 2}
```

就会把 作者 1，学术团体 1，地址 1 和 作者 2，学术团体 2，地址 2 分成两部分，分别一行行的居中排列。而且这相邻的两块在标题页上也是居中的。

如果不想把作者名左右相邻排版，也可以把它们上下排列，这时只要用 `\\` 取代 `\and` 就可以了。此时，可以用紧接 `\\` 后的可以省略的长度参数 [距离] 来调整竖直距离。

如果没有给出 `\date` 命令，就会自动在标题页的作者项后加上当前日期。另一方面，命令 `\date{日期文本}` 就会在出现日期的地方显示 日期文本。这个地方可以插入任何文本，也可以用断行命令 `\\` 以插入多行文本。

命令

```
\thanks{ 脚注文本 }
```

可以出现在 标题、作者和日期文本的任何地方。它会在命令出现的地方加上一个标志，而在标题页上以脚注的形式排版 脚注文本。

要想利用出现在 `\title`, `\author`, `\date` 和 `\thanks` 中的文本生成标题页, 就需要调用

`\maketitle`

命令。标题页本身没有页码, 后续文档第一页的页码为 1. (对于 `book` 类, 页码是由 3.3.5 节中的特殊命令控制的。) 只有在 `book` 和 `report` 类中, 标题页才会单独占一页。而在 `article` 中, 当使用了命令 `\maketitle` 时, 就会用来自于 `\title`, `\author` 以及可能的 `\date` 和 `\thanks` 中的条目在第一页上创建居中标题头。如果用了 `titlepage` 文档选项, 那么即使在 `article` 类中也会让标题位于单独一页上。

对于在 `titlepage` 环境中的没有格式的标题页, 命令 `\title` 和 `\author` 是没有作用的, 整个标题页的设计来自于作者在这个环境中的定义。此时我们可以利用第四章中的所有构造命令。在这种情况下, 当结束 `titlepage` 标题页环境时, 就会完成标题页的排版, 因此不再需要调用 `\maketitle` 命令。

3.3.2 摘要

可以用下面的命令生成摘要:

`\begin{abstract}` 摘要文本 `\end{abstract}`

在文档类 `report` 中, 摘要是位于单独一页上的, 而且有页码; 在 `article` 中, 摘要紧接标题头位于第一页上。但是如果选择了 `titlepage` 文档类选项, 摘要也是单独占一页的。在文档类 `book` 中没有摘要。

3.3.3 章节

下面的命令可以用来自动生成有序的章节:

`\part` `\chapter` `\subsection` `\paragraph`
 `\section` `\subsubsection` `\subparagraph`

除 `\part` 外, 其他命令形成一个章节序列。在文档类 `book` 和 `report` 中, 最高的章节层次是 `\chapter`。每一章可以用 `\section` 命令分为节, 每一节又可以进一步用 `\subsection` 等等分成更小的节。在文档类 `article` 中, 章节序列是由 `\section` 开始的, 因为这时不能用 `\chapter` 命令。

所有这些命令的语法是

`\章节命令 [短标题]{ 标题 }` 或者
`\章节命令 *{ 标题 }`

在第一种情形中, 把序列中下一个编号赋给章节, 并把这个编号同来自于 标题 的文本一起做为章节标题显示出来。短标题 文本用于目录 (3.4 节) 和页眉 (假定已选择了 `headings` 页面样式) 的显示。如果没有这部分文本, `LATEX` 就假定其内容与 标题 是一样的。通常不需要给出 短标题, 除非章节的标题太长, 不适于出现在目录和页眉中。

在第二种 (*- 形式) 情形中, 不会显示出章节编号, 而且也不会出现在目录表中列出该项 (见 3.4.3 节中的例外)。

章节标题的大小和编号的长度与章节命令在序列中的位置有关。对于文档类 `article`,

`\section` 命令就生成单个数字 (如 7), 而 `\subsection` 命令生成两个数字, 中间用点号分开 (如 7.3), 其他依次类推。

在文档类 `book` 和 `report` 中, 用 `\chapter` 命令给出的章标题中以单个数字作编号, 而 `\section` 得到的则是两个数字, 依次类推。而且, `\chapter` 命令总是开始新页, 并在章节标题上面打印 **Chapter n** , 这里的 n 表示当前的章编号。(注意, 在 CCT 中文 L^AT_EX 中, 对这里的英文显示进行了汉化, 见第三部分。)在本书当前位置, 我们是在 *Chapter 3, Section 3.3, Subsection 3.3.3*。

对于每个章节命令, 都有一个内部计数器, 每当调用一次命令, 对应的计数器就会增一, 而当调用下一个更高级的章节命令时, 就会把它重新设置为 0。*- 形式的章节命令不改变这些计数器的值, 这个事实在有些情况下会造成一种混乱, 例如当标准形式和 *- 形式的章节命令混合使用, 而 *- 形式命令在序列中的位置要比标准形式的高时, 就会出现这种问题。然而, 如果 *- 形式的命令总是低于标准形式的命令, 这就不会产生任何问题。下面的序列

```
\section ... \subsection ... \subsubsection*
```

的结果就是 `\section` 和 `\subsection` 有编号, 而 `\subsubsection` 没有编号。

章节命令 `\part` 对计数器的处理比较特殊, 它对其他章节命令的编号没有影响。

章节的自动编号就意味着在写作的时候, 可以不需要知道编号。作者没有必要按最终的顺序进行创作, 可以随时加进或去掉一些章节。如果作者在正文中想引用某一章节的编号, 那么这就需要一种间接输入编号的机制。将在 8.3.1 节描述 L^AT_EX 的交叉引用系统, 它利用如下两条基本命令完成这个任务:

```
\label{名称} \ref{名称}
```

前一条命令给章节编号赋一个名称 (也称为关键词), 而后一条命令则是用在正文中, 以引用章节的编号。关键词名称可以是字母、数字或符号的任意组合。例如, 本书中在 8.3.1 节开头使用了命令 `\label{sec:xref}`, 因此上面这句话中就只需包含进命令 `\ref{sec:xref}`。

还有一条引用命令, 那就是 `\pageref`, 它显示对应的 `\label` 命令所在那页的页码。

引用命令还可以用在许多其他地方, 前提条件只要那些项的编号是自动生成的, 例如我们可以引用图、表和公式的编号。

每一个章节命令都有一个层次号, 如 `\section` 总是第 1 层, `\subsection` 为第 2 层, ..., `\subparagraph` 为第 5 层。在文档类 `article` 中, `\part` 为第 0 层, 而在 `book` 和 `report` 类中, `\part` 为第 -1 层, `\chapter` 成为第 0 层。章节编号向下进行的层次由 `secnumdepth` 决定。对于 `book` 和 `report`, 其值为 2, 而对于 `article`, 其值为 3。这也就是说对于 `book` 和 `report`, 章节编号只进行到 `\subsection` 层次, 而对于 `article`, 则是到 `\subsubsection`。

为了拓展 (或减小) 章节编号的层次, 那就必须改变 `secnumdepth` 的值。这可以用命令

```
\setcounter{secnumdepth}{数}
```

来实现。(在 7.1 节中对 `\setcounter` 命令进行了解释。)在 `article` 类中, `secnumdepth` 可以取 -1 到 5 之间的值, 而对于 `book` 和 `report` 类, 可取 -2 到 5 之间的值。取最小

值或者更小的值意味着禁止对所有的章节进行编号。

在文档中也可以用如下命令来改变章节命令的初始值:

```
\setcounter{章节名称}{数}
```

这里的 **章节名称** 指的是没有前缀 \ 的章节命令名称。当要用 L^AT_EX 处理某一章节时, 这条命令是相当有用的。例如,

```
\setcounter{chapter}{2}
```

就把 \chapter 计数器设为 2。当下一次调用 \chapter 时, 其值会增加 1, 即得到 Chapter 3。

有时候需要改变章节标题的字体大小和样式。这可以用在 4.1.2~4.1.6 节中描述的用于章节标题文本的声明达此目的。例如,

```
\section*{\Large\slshape Larger Slanted Font}
```

就会以 \Large 尺寸和 \slshape 样式显示出节标题 ‘Larger Slanted Font’。我们不推荐使用这种抛弃标准方式的做法, 因为这些修改只对标题文本有作用, 而对其前面的编号没有影响。

3.3.4 附录

可以用 \appendix 命令来加进附录。这一命令具有重新设置 article 类中的节计数器和 book 与 report 类中章计数器的作用, 而且它把这些章节命令的编号形式从数字变为大写字母 A, B, ...。另外, 还自动用单词 ‘Appendix’(或者相应语言中的对应词) 代替 ‘Chapter’, 因此后续章节的标题前缀变为 ‘Appendix A’, ‘Appendix B’, 等等。低层章节命令的编号中用字母取代了章节编号, 如 A.2.1。另外附录也可以包含在如下一个环境中:

```
\begin{appendix}
```

附录文本

```
\end{appendix}
```

3.3.5 书的结构

为了简化书的结构, 在 book 类中提供了命令

```
\frontmatter
```

前言, 目录

```
\mainmatter
```

正文的主体

```
\backmatter
```

参考文献, 索引, 版本记录

\frontmatter 命令把页码切换为罗马数字格式, 并且不对章进行编号; 而 \mainmatter 命令把页码重设为阿拉伯数字 1, 并激活对章的编号; 而在 \backmatter 中又再次停止对章的编号。

3.4 目录表

3.4.1 自动条目

L^AT_EX 可以自动为整篇文档准备和显示一个目录表。这个表包含章节号和相应的标准形式的章节命令中的标题，以及章节起始页码。出现在目录表中的章节深度可以在导言中用命令

```
\setcounter{tocdepth}{数}
```

来设定。这里的 数 与前面描述的 secnumdepth 计数器中的意义相同，利用后者可以固定自动分章节的最深层次。在默认方式中，目录表中包含章节的最深层次与自动章节编号的最深层次是一样的，也就是说，对 book 和 report 类，最深层次为 \subsection，对 article 类，则到 \subsubsection。

3.4.2 显示目录表

目录表的生成和显示是用下面的命令实现的：

```
\tableofcontents
```

把这条命令放在希望目录出现的地方，通常就放在标题页和摘要的后面。

这就导致了一个两难的处境，因为目录表的信息是显示在靠文档开头的地方，而这些信息直到文档结束才可能全知道。L^AT_EX 是如下解决这个问题的：当文档第一次被处理时，并没有包含进任何目录表信息，而是由 L^AT_EX 打开一个与文档文件同名、而后缀为 .toc 的新文件；在后面的处理过程中，把目录表所需的条目写到这个文件中。

当下次用 L^AT_EX 处理这个文档时，\tableofcontents 命令读入与文档文件的基本名相同的 .toc 文件，从而显示出目录表。当继续进行后面的处理时，如果在前一次运行后做了某些改变，那么 .toc 文件会被更新。而这时显示出来的目录表是对应于前一次文档的。正是由于这个原因，对于最后的文档，应该运行两次 L^AT_EX。

3.4.3 其他条目

*- 形式的章节命令不会自动加入到目录表中。为了把它或其他条目插入到目录表中，可以用命令：

```
\addcontentsline{toc}{章节名称}{条目文本}
```

```
\addtocontents{toc}{条目文本}
```

用第一条命令，就会形成目录表格式的相应条目，其中 section 标题头要比 chapter 的向里缩进一些，但比 subsection 缩进的少。这是由 章节名称 参数决定的，取值为没有前缀 \ 字符的章节命令（如 section）。条目文本 与页码一起插入到目录表中。如果想同标准形式那样也有章节编号，那么条目文本必须具有 \protect\numberline{章节名称}{文本} 的形式。

\addtocontents 命令可以把任何所希望的命令或文本加入到 .toc 文件中。它可以是一条格式化命令，例如 \protect\newpage，当显示目录表时就会执行这些命令。

实际上, 在 .toc 文件中, 相应于目录表中的每一条目, 都是一条 `\contentsline` 命令, 该命令的语法为

```
\contentsline{ 章节类型 }{\numberline{ 章节编号 } 标题文本 }{ 页码 }
```

当调用了 `\tableofcontents` 命令时, 就会读入这些命令。可以利用文本编辑器修改或插入这一命令。其中 章节类型 表示章节的层次, 例如 section, 而 章节编号 是它的编号, 例如 2.3, 页码 则是条目所在页的编号。

3.4.4 其他列表

除了目录表, 插图与表格的列表也可以由 L^AT_EX 自动生成和显示出来。生成这些列表的命令为:

```
\listoffigures    读和 (或) 生成文件 .lof
```

```
\listoftables     读和 (或) 生成文件 .lot
```

在这些列表中的条目是根据 figure 和 table 环境中的 `\caption` 命令自动生成的 (见 6.7.6 节)。其他条目可以用与目录表相同的命令生成, 它们的一般形式为:

```
\addcontentsline{ 类型 }{ 格式 }{ 条目 }
```

```
\addtocontents{ 类型 }{ 条目 }
```

这里的 类型 代表 toc(目录表)、lof(图的列表) 和 lot(表格的列表) 三种类型中的一种。格式 参数就是上面描述的目录表中的章节名称, 或者图列表中的 figure, 以及表格列表中的 table。条目 参数表示要插入相应文件中的文本。

3.5 精调正文

3.5.1 单词与字符的距离安排

单词和字符之间的距离, 通常都是由 T_EX 自动安排的, 它不但利用字符的自然宽度, 而且还考虑特定字符组合的变化。例如, 如果 A 后接 V, 结果并不是 AV, 而是 AV; 也就是说, 为了好看, 它们彼此稍移近了一点。在一行中单词之间的距离是一致的, 而且距离的选择要保证左右边界与页边界对齐。这也称为左右调整。T_EX 还尽力使不同行中单词间的距离尽可能一致。

由标点符号结尾的单词, 其后要给出一个额外的空白, 具体大小要看到底是哪种符号了: 接在句号 ‘.’ 或惊叹号 ‘!’ 后面的空白就要比接在逗号后面的大。这一点相应于英文的排版规则, 即在句子与句子之间的空白应该多一点。在某些情形中, 自动安排的结果可能不会令人满意, 这样就可以按下面几节中的方法进行调整。

句子结束与句号

T_EX 把接在小写字母后面的句号做为句子的结束符, 这时就会插入额外的单词间距。这样就有可能与缩写产生混淆, 例如 i. e., Prof. Jones 或 Phys. Rev., 中的句号, 此时需要的是正常单词间距。为此可以用字符 ~ 或 _ 取代通常的空格。(字符 _ 只是为了表示一个空格, 否则根本看不到它。) 这两种方法都是插入通常的单词间距; 而且 ~ 还禁止在

这一点断行。所以上面的例子应该输入 `i.~e.`, `Prof.~Jones` 和 `Phys.\ Rev.`, 以得到输出 `i. e.`, `Prof. Jones` 和 `Phys. Rev.`, 这时就得到了正常的字符间距。对前两种情况, 必要时可强迫其中的某些相邻字符必须在一行上, 而对于第三种情况, 把 `Phys.` 和 `Rev.` 放在两行上并没有什么不妥。

紧接在大写字母后的句号并不认为是句子的结束, 而只认为是一个缩写。可如果它真的是结束一个句子, 那就需要在句号前面加上 `\@`, 以得到额外的间距。例如, 句子 ‘this sentence ends with NASA.’ 就应该输入为 `this sentence ends with NASA\@`。

法语间距

可以用命令 `\frenchspacing` 来告诉 \TeX 不必增加某些情形中的额外单词间距, 当再用 `\nonfrenchspacing` 命令时, 就又恢复到原来的方式。对前一种情形, `\@` 命令是被忽略的, 可以不用。

字符组合 “和”

用命令 `\,` 可以生成一个很小的距离。在某些情况下, 这条命令是很有用的, 例如, 当双引号 “和” 与单引号 ‘和’ 在一起的时候, 可以用 `\,` 把它们分开。文本 ‘`\, ‘Beginning’ and ‘End’\,`’ 的输出为 “‘Beginning’ and ‘End’”。

倾斜校正

当字体从斜体 (意大利斜体或 *slanted*) 变到竖直字体时, 由于最后一个字符会斜穿进接下来的空白中, 从而使得空白间距显得比原来要小。因此必须在这儿加上视不同字符而不同的额外间距 (*d* 的要比 *a* 的大)。 \TeX 用命令 `\/` 来加进这个额外间距, 称之为 倾斜校正。当一个倾斜字母后接一个竖直字母时, 总要用这种校正, 例如 `{\slshape slanted\/} spacing` 可以得到 *slanted spacing*。当倾斜字母后接逗号或句号时, 这种校正可以不加。

\LaTeX 2_ϵ 的字体命令 `\textsl` 和 `\textit` (4.1.4 节) 会自动加上倾斜校正, 这也是它比字体声明 `\slshape` 和 `\itshape` 好用的另一个优点。例如, 不要用 `{\slshape slanted\/}`, 而应该用 `\textsl{slanted}`。然而, 有时候我们可能希望没有这种校正, 那么可以在倾斜字母后面用 `\nocorr` 命令以达此目的。

取消连写

`\/` 命令也可以用来禁止连写。所谓连写, 就是特定的字母组合作为一个字符印刷。文本 `shelf\/ful` 的结果是 *shelfful*, 而不是 *shelfful*。在 2.6.8 节中已提到, \TeX 把字母组合 *ff*, *fi*, *fl*, *ffi* 和 *ffl* 处理成连写。输入 `f\/f\/i`, 可以得到 *ffi*, 而不是 *ffi* 连写。

`\/` 命令也可以用来取消特定字母组合之间的距离减少, 如 *AV* 和 *Te*:

`A\/V` *AV* `T\/e` *Te* 而不是 *AV Te*。

插入任意距离

可以用下面的命令在文本间插入任意大小的距离:

`\hspace{ 距离 }`

`\hspace*{ 距离 }`

这里的 距离 就是所要指定的间隔长度, 例如 1.5cm 或 3em。(注意一个 em 就是当前字样中字母 M 的宽度。)

这两条命令在调用点和接下来要显示的对象之间插入宽度等于上述 距离 的空白。用标准形式 (没有 *) , 可以使得, 若间隔恰好位于两行之间时, 就去掉这个空白, 这就如同在一行的开头, 空格要去掉一样。而另一方面, * 形式不管任何情况, 都会插入空白。

这里定义的距离也可以是负数, 这样该命令就如同退格, 在另一个字符上显示新的字符。

在这条命令前后的空格仍然有作用:

```
This is\hspace{1cm}1cm   This is      1cm
This is\hspace{1cm}1cm   This is      1cm
This is\hspace{1cm} 1cm   This is      1cm
```

命令 `\hfill` 就是 `\hspace{\fill}` 的缩写形式 (见 2.5.2 节)。它会在其两边的文本之间插入足够大的空白, 从而使它们分别靠近左右页边。Left `\hfill` Right 的输出就是

```
Left                                                                    Right
```

在一行上有多个 `\hfill`, 那个它们插入的空白宽度都相同, 从而使这一行变得左右协调。例如, 输入 Left `\hfill` Center `\hfill` Right 的结果为

```
Left                                                                    Center                                                                    Right
```

如果 `\hfill` 位于一行的开头, 根据 `\hspace` 标准形式的行为准则, 这个空白是被去掉的。如果确实需要在一行的开头或结尾加上一个弹性空白, 那就必须用 `\hspace*{\fill}`。然而, L^AT_EX 还提供许多命令和环境, 以简化许多这种情形的应用 (见 4.2.2 节)。

在 L^AT_EX 中, 还有如下一些长度固定的水平距离:

`\quad` 和 `\qquad`

命令 `\quad` 插入一个长度等于当前字体尺寸的空白, 即若当前为 10pt 字体, 则插入 10pt, 而 `\qquad` 是其两倍。

插入长度可变的 和 _____ 序列

还有两条命令, 其用法同 `\hfill` 完全一样:

`\dotfill` 和 `\hrulefill`

当不想插入空白时, 可以用上述命令如下插入点或直线:

```
Start \dotfill\ Finish\\
Left \hrulefill\ Center \hrulefill\ Right\\
```

的结果为

```
Start ..... Finish
Left _____ Center _____ Right
```

可以在一行上出现 `\hfill`, `\dotfill` 和 `\hrulefill` 的任意组合。如果在同一地方多次使用这种命令, 那么相应的填充就比单个时的大很多。

```
Departure \dotfill\dotfill\dotfill\ 8:30 \hfill\hfill
Arrival \hrulefill\ 11:45\\
```

结果为

```
Departure ..... 8:30                      Arrival _____ 11:45
```

3.5.2 断行

把文本断成行的操作是由 TeX 和 LaTeX 自动进行的。然而，也有这样的时刻，你想强迫或希望在某些地方断行，或不想在某些地方断行。

命令 \\

可以用 \\ 命令来得到一个有或没有额外行距的新行。该命令的语法为

```
\\[ 距离 ]
```

```
\\*[ 距离 ]
```

这里的可省参数 距离 是一个长度，它用来定义增加多少额外的行间距。如果此时需要开始一个新页，那就不加这个额外间距，新页开头为下面的文本行。*- 形式禁止在两行之间分页。

用 *[10cm]，当前行就会结束，在这行与后面文本加上一个 10cm 的竖直距离，而且这两行文本必须在同一页上。如果需要一个分页，那就会在当前行前面进行，这样当前行，10cm 的竖直距离，以及后面的文本就位于新页上。

命令 \newline 与没有可省参数的 \\ 命令是一样的。也就是说，新行之前没有额外的空白，而且此处也可以分页。

这两条命令都可以用在段落中间，除此而外，它们再没有其他意义。

其他的断行命令

命令 \linebreak 用来鼓励或强迫在文本中某点断行。它的形式为

```
\linebreak[ 数 ]
```

这里的 数 为可省参数，它是一个介于 0 到 4 之间的数，用来表示断行的重要性。这条命令鼓励断行，数越大，鼓励的力度就越大。取 0 值表示允许在本来不会断行的某处（如一个单词的中间）断行，而 4 意味着必须断行，它等同于没有参数的 \linebreak。这条命令与 \\ 或 \newline 的区别在于当前行要进行左右调整，增加单词间距，以使文本充满该行。而对于 \\ 和 \newline，当前行最后一个单词后以空白填充，单词间距保持正常值不变。

与之相反的命令为

```
\nolinebreak[ 数 ]
```

它不鼓励在给定点断行，这里的 数 表示不鼓励的力度。而且没有参数的命令 \nolinebreak 同 \nolinebreak[4] 有一样的效果，即在这里绝对不可能进行断行。

另外一种使文本呆在一行上的方法是用命令 \mbox{文本}。对于类似于 ‘Voyager-1’ 这样的表达式，如果想在连字符处禁止断行，用这个命令就是非常方便的。

3.5.3 段落间距

段落间的正常间距是由长度 `\parskip`(3.2.4 节) 设置的, 可以用命令 `\setlength` 来改变其初始值:

```
\setlength{\parskip}{距离}
```

也可以用如下命令在某一特定段落间加上额外的竖直距离:

```
\vspace{距离}
```

```
\vspace*{距离}
```

即使在此处开始一个新页, 或者命令位于新页的开头, `*`-形式也要加上这个额外的竖直距离, 而同样对这两种情形, 标准形式则禁止加上额外的竖直距离。

如果在一个段落中间用这两条命令, 那就在当前行后面加上额外间距, 而且按通常那样对当前行进行左右调整。

距离参数可以为负数, 这样可以抬高后面文本的相应于正常显示的位置。

命令 `\vfill` 是 `\vspace{\fill}`(见 2.5.2 节) 的缩写形式。其等价于水平间距的 `\hfill`, 它会插入足够的竖直空白, 以使得文本的顶部和底部与页面的顶部和底部对齐。对于同时出现多个 `\hfill` 的解释, 也同样适用于 `\vfill`。如果这条命令位于一页的开头, 它没有任何作用, 这同标准形式 `\vspace{\fill}` 一样。如果想在页面底部加上一个弹性间距, 那就必须用 `*`-形式 `\vspace*{\fill}` 了。

其他的用于增加段落间距的命令是

```
\bigskip \medskip \smallskip
```

这些命令根据在文档类中声明的字体尺寸来确定增加的竖直距离。可以用 `\bigskipamount`, `\medskipamount` 和 `\smallskipamount` 重定义其大小, 例如,

```
\setlength{\bigskipamount}{5ex plus1.5ex minus2ex}
```

如果不用空行, 也可以用命令 `\par` 来表明一个段落的结束。

3.5.4 段落的缩进

段落缩进的大小是由长度 `\parindent`(见 3.2.4 节) 决定的, 可以用命令 `\setlength` 改变它的值:

```
\setlength{\parindent}{距离}
```

这样每段的第一行左边都有一个宽度为上面 距离 参数指定的空白。如果想取消某一段落的缩进, 或者如果没有缩进 (如一节的第一段) 时想强迫出现缩进时, 可以用下面的命令达此目的:

```
\noindent 和 \indent
```

但是注意要把它们放在段落的开始, 这样它们才会发挥作用。

3.5.5 分页

在 $\text{T}_{\text{E}}\text{X}$ 和 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 中, 同自动断行一样, 把文本分成页的操作也是自动进行的。同样的, 我们也可以借助于一些命令, 来影响程序确定何处为分页点。

正常分页

命令

`\pagebreak[数]`

`\nopagebreak[数]`

等价于断行中的 `\linebreak` 和 `\nolinebreak`。如果 `\pagebreak` 位于两段之间，那就会在此处进行分页。如果它位于一段的中间，那么就会在当前行完成后进行分页。同样要对该行进行左右调整。

命令 `\nopagebreak` 具有相反的作用。当位于在两段之间时，它禁止在此处分页；当位于一段的中间时，那它禁止可能在当前行后面出现的分页。

介于 0 到 4 之间的可省略参数表达了鼓励或不鼓励分页的力度。而且 `\linebreak` 要做更多的事，它除了给断开的行增加单词间距，进行左右调整，还要增加行间距，以使页面顶部和底部分布一致。

要想在一页中间结束该页，并用空白填充余下空间，再开始一个新页，那么可以调用如下命令

`\newpage`

它等价于分页中的 `\newline` 命令。

有图表时的分页

如果文本中有表格，插图或者为图形预留的空间时，它们会被插入在相应命令出现的地方，但前提条件是在当前页上要有足够的空间。如果没有足够的空间，就会继续排版正文，而把图表保留到下一页上。

命令

`\clearpage`

同 `\newpage` 命令一样结束当前页，而且用一页或多页来输出没有显示的图表（请见 6.7 节）。

两列模式时的分页

如果选择了 `twocolumn` 文档类选项，或者用了 `\twocolumn` 命令，那么 `\pagebreak` 和 `\newpage` 命令就会结束当前列，开始一个新列，即把列当做页处理。另一方面，`\clearpage` 和 `\cleardoublepage`（见下面）就是中止当前页，如果必要的话，插入一个空列。

两面模式时的分页

当选择了 `twoside` 文档类选项时，还可以用一个分页命令：

`\cleardoublepage`

它的作用同 `\clearpage` 一样（中止当前页，使所有未显示的图表输出在后面的页上），而且接下来的页应是奇数页。如果必要的话，由此会得到一个具有偶数页码的空页。

有限制的分页

声明 `\samepage` 可以使得分页只会发生在段落之间, 单独公式或文本的前后, 一个章节标题的前后, 或者列表环境中的 `\item`(第一个除外) 之间。如果想在正文其他地方进行分页, 那就必须用 `\pagebreak` 命令了。

实际上这一功能的作用并不像期望的那样好, 正是由于这个原因, \LaTeX 2_ϵ 提供了一些新方法, 以控制不必要的分页。只是为了与原来文档兼容, 才继续保留了 `\samepage` 命令。

控制分页

\LaTeX 2_ϵ 用如下命令提供了稍微增加当前页高度的可能性:

```
\enlargethispage{尺寸}
```

```
\enlargethispage*{尺寸}
```

这两条命令把定义中的 `尺寸` 只加到当前页的 `\textheight` 上。有时候只需要稍微一点儿的调整, 就可以避免一个糟糕的分页。*- 形式的命令在当前页上则无论如何也要调整行距, 以最大化文本高度。

分页的其他方法

在 3.2.5 节中已提到, 在文档类 `book` 或选项 `twoside` 中, 在每一页上, 页面底边的位置是一样的, 而在其他情形中则不然。这意味着在这两种情形里, 需要在适当的位置(如段落之间)上插入行间距。

这种情形是由于在 `book` 类和 `twoside` 选项中, 使用了 `\flushbottom` 声明, 而在其他所有的情形中, 则使用的是 `\raggedbottom` 声明。它们最初是在类和选项的内部定义中设置的, 但是用户也可以在导言中或正文的其他地方用这些命令。除非遇到相反的命令, 或当前环境结束, 否则该命令一直有效。

通常不要通过选择页面长度 `\textheight` (3.2.5 节) 来在一页上放更多的文本。然而, 如果确实需要多挤进一行或两行文本, 那就必须为它们清除竖直间距。这可以通过放在适当位置的带负长度参数的 `\vspace` 或 `\` 命令来做到。最佳地方是在单独公式、枚举、列表、表格或插图的前后(见第四章)。

我们在这里不会去详细讲解分页的内部规则, 只提及下面一点, 对那些不鼓励在此分页的点, \TeX 要加上所谓的罚点(对应于命令 `\penalty`), 而负的罚点则意味着鼓励分页。段落中的每行都对应着一个罚点, 比如说是 10 的一个罚点 (`\linepenalty=10`), 这样就可以使得在段落中间分页比段落之间分页要难。

段落的第一行和最后一行赋给 150 的一个罚点(也就是说进行了 `\clubpenalty=150` 和 `\widowpenalty=150` 的赋值), 从而不鼓励(但没有禁止)在一页上只留下段落中的一行。

因此, 做为一种手段, 我们也可以通过修改罚点值, 以获取所需的分页。例如, 若取 `\clubpenalty=450`, 那么在段落中第一行后分页就要比平常时的难上三倍。把罚点值取为 10000 或更多, 就会使得绝对不会在此处分页。然而对罚点的修改, 只应该用在其

他方法都行不通的时候，因为这破坏了与其他部分文档之间的平衡，从而可能引起更多古怪的分页。

3.6 断 词

当要对一行进行左右调整时，有时候会出现无法在单词之间断行的情形，因为这样做要么会使单词挤得很紧，要么彼此之间离得很远。这样就需要分开一个单词。这一重大的任务是由 \TeX (即 \LaTeX 的本质基础) 来完成的，它用的断词算法对英语 (绝大多数作者用的都是这种语言) 文本 (绝大多数情形下) 处理的很好，然而，它也有出错的时候，这就需要我们干预它了。

如果通常的 \TeX / \LaTeX 用于其他语言，或者在英文中出现了外文单词，也很有可能出现不正确的断词。(在 3.6.5 和 3.6.6 节有关于 \LaTeX 与其他语言共用的更多讨论。) 在这种情况下，那也必须用如下的方法重定义 \TeX 的断词算法。

3.6.1 人工断词

最简单地纠正一个被错误断开的单词的方法就是在这个单词中间的恰当位置上插入命令 $\backslash-$ 。例如对于单词 `manuscript`， \TeX 的断词算法无法把它断开，因此如果在断行时它造成了一些问题，就可把它写成 `man\-\u\-\script`。这告诉 \TeX ，在必要时可以把它断成 `man-uscript` 或 `manu-script`，而不用顾及通常的规则。

$\backslash-$ 只是使得在指定位置有可能断开，而没有强迫在此处断开。如果作者坚持要在某一点断开某个单词，比如说在 `manuscript` 的 `u` 和 `s` 之间，那可以输入 `manu-\linebreak script` 来做到。但是，我们并不推荐这种粗暴的做法，因为不管以后对正文有没有修改，这个地方总是有个断开点。

对于英文，单词的拼写即使断开也是不变的，而对于其他语言，这可就不一定了。例如在德语中，就有一条规则，如果 `ck` 被分开，它就变成 `k-k`。在 \TeX 中可以用一般性断开命令来做到这一点：

```
\discretionary{前}{后}{无}
```

这里的 `前` 与 `后` 都表示字母 (有连字符)，它们指的是如果要断开的话，在断点两边的字母，而 `无` 则是没有断开时的正常写法。因此 Boris Becker 的姓名应该写成

```
Boris Be\discretionary{k-}{k}{ck}er
```

只有在特殊情形下才需要这种输入。顺带提一下， $\backslash-$ 命令的定义就是

```
\discretionary{-}{}{}。
```

3.6.2 断法列表

断开不正确，但是又经常在正文中出现的单词可以放在导言的一个例外列表中，以避免每次费事地插入 $\backslash-$ ：

```
\hyphenation{列表}
```

这里的 `列表` 就是一组单词，用空格或换行分开，而且用连字符表示可允许的断点。例如，


```
\hyphenation{man-u-script com-pu-ter gym-na-sium
              coun-try-man re-sus-ci-tate ...}
```

这个列表中可以包含通常的从 a 到 z 的所有字母，但不能出现特殊字符或重音。

3.6.3 禁止断词

另外一种避免糟糕断词的做法就是关闭一两个段落的断词功能，实际上

```
\begin{sloppypar} 段落文本 \end{sloppypar}
```

环境并没有禁止断词，只是允许更大的单词间隔，而不会给出警告信息。这就意味着实际上所有行都是在单词间断开的。也可以在导言中或当前环境中加入命令 `\sloppy` 来减少整篇文档或当前环境中的被断开的单词数目。当行相当窄时，推荐使用这种方法。

当命令 `\sloppy` 起作用时，那么可以用如下环境来暂时重新打开断词：

```
\begin{fussypar} 段落文本 \end{fussypar}
```

在环境中用命令 `\fussy` 也可以得到相同的效果。

3.6.4 行宽与断词

现在有必要在这里补充一些关于行宽与断词的注解。

当 \TeX 处理到一段末尾时，它就尝试在考虑 3.5.1 节中提到的间距要求的前提下，通过在单词间断行，把文本组织成长度相等的行；这也就是说，不会出现断词。如果这种尝试行不通，那它就开始尝试在单词中间断开。当行宽很大，或者单词的平均长度很小时，在单词之间断行是相对比较容易的。而且，对于给定的行宽，如果字体尺寸是小的，那么被断开的单词数目也会少些。

若行宽很窄，即使可以在单词中断开，可能左右调整也很困难，这样 \TeX 就需要插入比正常情况下所允许的多一些的单词间距。要做到这一点，可以用上面给出的 `sloppypar` 环境或 `\sloppy` 命令。这样可以进行调整，但是代价是在单词间加进了不可接受的间距。在这两种情形中， \TeX 都会显示出一条警告信息，要么是 `Overfull \hbox` (不可能左右对齐，一个单词将向右边突出) 或 `Underfull \hbox` (单词间距太大)。如果无法容忍这两种情形，那就必须对正文进行调整，利用命令 `\linebreak` 和 `\hfill` 来强迫出现适当的断词点。

3.6.5 其他与断词有关的内容

\TeX 用来进行断词的信息保存在一个名为 `hyphen.tex` 的内部文件中，其中包含一组供特殊断词算法所用的字母组合。这个列表是基于所考虑语言的统计研究结果的。这个文件中也包含一些例外列表，其中单词都是算法不能正确断开的。 \TeX 首先在这个例外列表中搜索，根据在其中可以找到的模式进行断词；如果在其中没有找到，那它就用自己的算法来断词。对于其他语言，那就必须给出不同的字母组合列表和例外列表了。

在 `hyphen.tex` 文件的尾部可以找到例外列表，其由前面给出的命令 `\hyphenation` 组成。这个列表中的每一项都可用文本编辑器进行编辑。

每次对断词文件进行改变，就必须生成一个新的格式文件。这是一种针对于 \LaTeX 的主要宏定义文件，它以可快速读入的紧致方式存贮。

顺便提一下，你可以用下面的命令在屏幕上查看 TeX 断词算法的结果：

```
\showhyphens{ 单词表 }
```

这会在屏幕上显示出 TeX 对 单词表 中所有单词的断开结果。例如，给出：

```
\showhyphens{interrelations penumbra summation}
```

那在屏幕上就会显示出

```
in-ter-re-la-tions penum-bra sum-ma-tion
```

3.6.6 多语言文本中的断词

在 TeX2.99 或更低的版本中，plain.fmt 或 lplain.fmt 中只允许有一组用于断词的字母组合。这些版本的 L^ATeX 程序可适用于英文或者另一种语言。在一个文档中不可能把多种语言的断词算法混用。

从 TeX 的 3.0 版本起，在格式中可以包含多个断词列表，这样就可以在一个文档中切换到不同的断词框架中，所使用的就是 TeX 新提供的 \language 命令。这一命令也可以用在针对于特定语言对 L^ATeX 进行的改编中，从而在输出中对某些英文单词进行翻译，并简化重音和标点符号的输入，以及修改 \today 等日期命令的定义。

第四章 文本排版

有许多方法可用来显示或强调文本，例如改变字体样式或字体尺寸，使文本居中、缩进等等。L^AT_EX 为我们提供了实现这些最常见的显示形式的命令。另外，还可以利用许多环境，进行文本排版。

4.1 改变字体

在排版印刷中，一组特定大小和样式的字母、数字及符号的组合称为一种字体。在 L^AT_EX 中用于正文的是一种直立的罗马字体，由文档开头处的 `\documentclass` 来定义其平均权重。可用的基本尺寸是 10, 11 或 12pt，分别相应于尺寸选项 10pt(默认值), 11pt 或 12pt。(注意一英寸大约等于 72.27 点，而一厘米大约等于 28.45 点。) 小括号 () 在纵向上占据的尺寸等于字体尺寸的高度和深度。

三种标准尺寸的视觉效果相应于这三个数字的比率而言是相当大的：

This is an example of the 10 pt font. ()

And this is the 11pt font for comparison. ()

And finally this is a sample of 12 pt font. ()

4.1.1 强调

在用打字机打出来的手稿中，强调文本的最简单方法就是用 下划线。在印刷时，制版工人通常要把用下划线的文本转化为斜体。在 L^AT_EX 中要从标准文本切换为强调形式，只需要用声明 `\em` 或命令 `\emph`。

声明 `\em` 的作用同下面将要讲解的其他字体声明一样，它改变当前字体，直到被其他相应的声明(它可以是 `\em` 自身)取消，或者当前环境结束为止(2.3 节)。也可以只用一对大括号 `{...}` 来创建一个环境。这是一种最简单的强调一小段文本的方法，请见示例：

This is the easiest way to `{\em emphasize}` short ...

This is the easiest way to *emphasize* short ...

在 L^AT_EX 2_ε 中新增加了 `\emph` 命令，这是一种强调一两个单词的更合逻辑的方法：

A more logical method of `\emph{emphasizing}` a word ...

A more logical method of *emphasizing* a word ...

注意它们的差别，声明的作用是当局部环境用右大括号结束时而终止，而命令只对其包含在大括号内的参数有作用。另外一种更精细的差别就是命令 `\emph` 自动插入倾斜校正(见 3.5.1 节)，而在声明 `\em` 中则必须人工加入这一校正。

声明和命令都切换进入一种强调字体。这也就是说如果当前字体是直立的，那就切换为斜体，而如果当前字体是 slanted，则就切换为一种直立字体。

可以嵌套强调，而且其非常易懂：

The {\em first}, second, and {\em third font switch}

The {\em first, {\em second, and {\em third font switch}}}

的结果都是 ‘The *first*, second, and *third font switch*’.

4.1.2 字体尺寸的选择

在 L^AT_EX 中可以用下面的声明来改变字体尺寸:

<code>\tiny</code>	<code>smallest</code>	<code>\Large</code>	larger
<code>\scriptsize</code>	<code>very small</code>	<code>\LARGE</code>	even larger
<code>\footnotesize</code>	<code>smaller</code>	<code>\huge</code>	still larger
<code>\small</code>	<code>small</code>	<code>\Huge</code>	largest
<code>\normalsize</code>	<code>normal</code>		
<code>\large</code>	<code>large</code>		

上面所有的声明都是相对于在文档类选项中的标准尺寸而言的。本书的标准尺寸为 10pt, 可以用 `\normalsize` 来选择该尺寸。

字体尺寸声明与所有其他声明的行为是一样的: 它立即发挥作用, 直到被另一个相反的声明所取消, 或者当前环境结束。如果在大括号 `{...}` 内调用, 声明的作用就只局限于大括号内, 这就如同一个无名环境:

```
normal {\large large \Large larger} normal again
normal large larger normal again
```

在 L^AT_EX 2.09 中字体尺寸声明也同时把所有其他字体属性 (4.1.3 节) 重设为默认值, 也就是平均权重的直立罗马字体。与之相反, 在 L^AT_EX 2_ε 中, 这些属性是不变的。试做一比较:

```
(LATEX 2.09) \sl slanted {\Large larger} slanted larger
(LATEX 2ε) \sl slanted {\Large larger} slanted larger
```

如果文档开头用的不是 `\documentclass`, 而是 `\documentstyle`, 那么这些尺寸声明的作用就与 L^AT_EX 2.09 中的相同, 这是为了与原来文档兼容才如此设计的。

用上面的命令来改变字体尺寸, 也同时自动改变了行间距。对于每一种字体尺寸, 都有一个对应的自然行间距 `\baselineskip`。可以在任何时候改变其值。例如, 如果自然行距是 12pt, 命令 `\setlength{\baselineskip}{15pt}` 就会把其增为 15pt。

在一段结束时有作用的 `\baselineskip` 命令被用来包装整段。这就意味着如果在一段中对 `\baselineskip` 进行了几次修改, 那么只有最后那次修改起作用。

每次修改字体的尺寸, `\baselineskip` 都会被重设为相应的自然行距。这样由命令 `\setlength` 所做的设置也就无效了。

要创建一个对所有字体尺寸都适用的行间距修改, 就必须用 `\baselinestretch` 因子, 其正常值为 1。实际上真正的行间距是

$$\text{\baselinestretch} \times \text{\baselineskip}$$

这样对所有的字体尺寸就会保持相应的间距。用户可以随时改变其值:

```
\renewcommand{\baselinestretch}{因子}
```

这里的因子是一个浮点数。取值 1.5 就意味着把行间距 (基线与基线之间的距离) 从相应于所有尺寸的自然间距增加了 50%。

然而, `\baselinestretch` 的新值只有当又进行了一次字体尺寸改变时才会发挥作用。为了在当前字体尺寸中实现一个新值, 就必须切换到一个新的字体尺寸, 然后马上切换回来。如果当前字体尺寸为 `\normalsize`, 那么序列

```
\small\normalsize
```

就可以达到所需要的效果。可以用任何类似的字体尺寸命令来取代 `\small`。

并不是在所有的尺寸中都可以用所有的字体样式。如果选择了一种不可能的字体尺寸和样式组合, 那么 \LaTeX 就会发出一个警告信息, 并在打印时告诉你用何种字体取代了该字体。

4.1.3 字体属性

字体的尺寸只是用来描述这种字体的几种属性中的一个。做为 \LaTeX 2_ϵ 内部集成的一部分, 在新字体选择框架 (NFSS) 中, 可以按 8.5 节所描述的那样, 用这些属性来选择字体。然而, 对普通用户来说, 就需要用声明和相应的命令来简化这一过程。

对于由 \TeX 和 \LaTeX 提供的计算机现代字体, 具有下列的属性和对应值:

Family(族) 指一般的综合样式。传统印刷中的族有 *baskerville*, *Bodoni*, *Times Roman*, *Helvetica* 等等的名称。标准 \LaTeX 2_ϵ 的安装中用如下的声明提供了三种族:

```
\rmfamily 切换 (回) 到一种罗马字体;
\ttfamily 切换到一个打字机 (typewriter) 字体;
\sffamily 切换到一个 sans serif 字体。
```

Shape(形状) 指的是字体的构成。在标准安装中可以使用的形状声明为

```
\upshape 切换 (回) 到一种直立字体;
\itshape 切换到一种斜体;
\slshape 选择一种叫 slanted 的字体;
\scshape 切换到小体大写字母 (SMALL CAPS)。
```

Series(序列) 指的是字体的宽度和权重 (黑度)。可用的声明为

```
\mdseries 切换 (回) 到平均权重;
\bfseries 切换到黑体。
```

上面并没有穷尽所有的字体属性, 但其涵盖了最标准的字体, 尤其是计算机现代字体。对于其他字体, 特别是 *PostScript* 字体, 还存在其他的属性。详情请见 8.5 节。

这些声明的用法同其他的一样, 通常包含在一对大括号 `{...}` 内, 如 `{\scshape Romeo and Juliet}`, 结果为 *ROMEO AND JULIET*。对于很长一段文本, 可以用相应的环境:

```
\begin{ 字体样式 }... 新字体中的文本 ...\end{ 字体样式 }
```

这样的开始和结束切换是非常清楚的。其中的 字体样式, 可以是上面的字体命令, 只要去掉 `\` 就可以了。

改变字体的一种属性, 而其他的保持不变, 就可以得到许多种组合。(然而, 这并非

意味着对每种可能的组合，都存在着一种字体与之对应；如果不存在的话，就会用另一种来替代。) 如果我们首先用 `\bfseries` 选择一种黑体序列，接着用 `\slshape` 把它改为 *slanted* 形状，那么我们就得到一种黑色的 *slanted* 字体：

```
normal and {\bfseries bold and
           {\slshape slanted} and back} again.
```

的结果为： `normal and bold and slanted and back again.`

最后要指出一点，声明 `\normalfont` 要把所有的属性（除了尺寸）重设成其默认值，即罗马字体，直立和平均权重。有时为了使字体有效，调用这条命令是非常有用的。

4.1.4 字体命令

对于上面所列的每一种字体声明，都有相应的字体命令，它们给其参数中的文本取指定属性的字体。

```
Family:  \textrm{ 文本 }      \texttt{ 文本 }  \textsf{ 文本 }
Shape:   \textup{ 文本 }      \textit{ 文本 }  \textsl{ 文本 }
        \textsc{ 文本 }
Series:  \textmd{ 文本 }      \textbf{ 文本 }
默认值:  \textnormal{ 文本 }
强调:    \emph{ 文本 }
```

注意这里也包含了 `\emph` 命令，其相应的声明是 `\em`。 `\textnormal` 的参数值要被设置成 `\normalfont` 所选择的标准字体。

这些改变字体的命令可以用来作用在一小段文本或单个单词上，这样就比在一个环境中放一个声明要合理得多。前面那个例子现在变为：

```
normal and \textbf{bold and \textsl{slanted}
and back} again.
```

结果仍然为： `normal and bold and slanted and back again.`

同 `\emph` 命令一样，当在直立和斜体 /*slanted* 字体之间切换时，这些字体命令会自动地加进必要的倾斜校正 (3.5.1 节)。当会出现这种校正时，可以利用 `\nocorr` 命令取消它，如

```
\textit{some italics\nocorr} without correction
{\slshape italics \nocorr\textup{without} correction}
```

请记住：字体属性声明和命令只适用于 $\text{\LaTeX} 2_{\epsilon}$ ，而不适用于 2.09 版本。

4.1.5 旧字体声明

为了与 $\text{\LaTeX} 2.09$ 兼容，仍旧保留了两字母的字体声明（它们很早就是 \TeX 的一部分）。

```
\rm Roman      \it Italic    \sc SMALL CAPS
\bf Bold face  \sl Slanted   \sf Sans Serif
\tt Typewriter
```

这些命令与新的相应命令相比有不同的作用方式，它们是严格地选择一种新字体，而不是只改变某一属性，但是不会改变当前尺寸。这就好像同时调用了 `\normalfont` 声明一样。

`{\bf text}` 等于 `{\normalfont\bfseries text}`

注意：在 \LaTeX 2_ϵ 之前的两个 NFSS 版本中，这些两字母声明被定义成新的长声明。为了处理用那些系统编写的文档，需要在导言中加进宏包 `newlfont`(8.8.3 节)。

4.1.6 其他字体

很可能你所用的 PC \TeX 软件具有不只上面所列的那些字体形状和尺寸。那么在 \LaTeX 文档中可以使用它们，方法是要么直接引用它们的名称，要么如果它是建立在 NFSS 之上的，可以使用它们的属性。

要用名称上载一种字体，可以用命令

`\newfont{\fnt}{名称 scaled 因子}` 或

`\newfont{\fnt 字体}{名称 at 尺寸}`

它把字体赋给新的字体命令名 `\fnt`。在前一种情形中，因子是一个整数，它是放缩因子的 1000 倍，这个因子用来对字体的基本或设计尺寸进行放大或缩小。在第二种情形中，字体被放缩到指定的尺寸。要安装一种 slanted, sans serif 字体，尺寸为 20.74pt，名称为 `\sss`，那么我们可以用命令

`\newfont{\sss}{cmssi17 at 20.74pt}`

上载 `cmssi17 at 20.74pt`。现在声明 `\sss` 就同 `\rm` 和 `\it` 改变字体一样把字体切换为上述字体，只是基线分隔没有改变。

与之相应的，也可以用属性来定义新字体声明（见 8.5.4 节）：

`\DeclareFixedFont{\sss}{OT1}{cmss}{m}{sl}{20.74}`

实际上，如果不知道当前所想用的字体编码和 `\sffamily`，或者不想那么精确地声明尺寸，那么也可以如下给出：

`\DeclareFixedFont{\sss}{\encodingdefault}{\sfdefault}`
`{m}{sl}{20}`

（默认值在 8.5.3 节解释。）

4.1.7 字符集与符号

\TeX 和 \LaTeX 显示所用的字符集是由程序实现的。对于不同的版本，约有 400 到 800 个的字体文件，分别相应于不同的字样、尺寸和放缩比例，每个文件由 128(将来会是 256) 个字符或符号组成。

每个字符集分别存贮在它们自己的文件中。在字符集中的每个符号都是通过一个介于 0 到 127(或者 255) 之间的数来编址的。命令

`\symbol{数}`

会生成当前字体中内部标识号等于 数的符号。在当前字体中符号 `l` 的内部编号为 62，因此可以用命令 `\symbol{62}` 打印出它来。标识号也可以用八进制（前缀 `'`）或十六进制（前

缀")数给出。因此符号命令 `\symbol{28}`, `\symbol{'34}` 与 `\symbol{"1C}` 是一样的, 都会生成 ϕ 。

`\symbol` 命令也可以用来生成还没有为它定义其他命令的符号, 例如,

```
{\tt\symbol{'40}\symbol{'42}\symbol{'134}}
```

得到 `"\`。

4.2 居中与缩进

4.2.1 居中文本

环境

```
\begin{center} 第一行 \\ 第二行 \\ ... \\ 第 n 行 \end{center}
```

把由 `\\` 命令分开的各节文本居中排列。(可以用 `\\[长度]` 来插入可以省略的额外行间距。)如果文本的长度超过一行, 就会用一致的单词间距把它分成几行, 除了最后一行外, 尽可能地使它充满每一行。不会有断词现象出现。

在一个环境内部, 可以用命令 `\centering` 来居中后接文本, 同样用 `\\` 做为行分隔符。声明的作用到该环境结束时为止。

单独一行文本可以把它做为 TeX 命令 `\centerline{文本}` 的参数来居中排列。

4.2.2 单边调整

环境

```
\begin{flushleft} 第一行 \\ 第二行 \\ ... \\ 第 n 行 \end{flushleft}
```

```
\begin{flushright} 第一行 \\ 第二行 \\ ... \\ 第 n 行 \end{flushright}
```

使得文本向左 (`flushleft`) 或向右 (`flushright`) 对齐。如果这一段文本在一行中放不下, 就把它断成相同单词间距的几行, 这同 `center` 环境一样。而且也不会出现断词。

也可以在环境内部用如下的声明达到同样的效果:

```
\raggedleft 代替 flushleft 环境, 而
```

```
\raggedright 代替 flushright 环境。
```

4.2.3 两边缩进

可以用下面的环境来使一段文本的两边都缩进一定的长度:

```
\begin{quote} 文本 \end{quote}
```

```
\begin{quotation} 文本 \end{quotation}
```

为了与通常的文本区分开, 在这段文本的上方和下方插入了额外的间距。

要显示的文本可具有任意的长度; 它可以是一句话, 也可以是一个段落, 甚至是几个段落。段落间仍然用空行表示分段, 但是在显示文本的开始和结束时并不需要输入空行, 因为无论如何这些地方总会自动地插入竖直间距。

上面两种引用环境的区别在于:

在 `quotation` 环境中，第一段的第一行具有缩进，而在 `quote` 环境中，则没有这种缩进，而是在环境的前后插入更多的竖直间距。

当前文本就是用 `quotation` 环境生成的，而本小节开头的文本则是用 `quote` 环境生成的。

为了全文排版风格上的一致性，只有当在通常的文本段落中第一行具有缩进时，我们才有必要采用 `quotation` 环境。

4.2.4 诗歌缩进

为了排版诗歌、韵文，需要两边都进行缩进，为此我们可以采用

```
\begin{verse} 诗 \end{verse}
```

环境。例如：

节与节之间用空行分开，

而每一节中用 `\\` 分开不同的行。

如果一行太长，以致于超过行宽，就会自动对它进行左右调整，并在下一行上继续，而且进一步向里缩进。

以上的规则描述就是用 `verse` 环境排版的。

上述的缩进框架可以嵌套使用。在一个 `quote` 环境中可以有其他的 `quote`, `verse` 或 `quotation` 环境。每嵌套一次，文本的两边都会加上额外的缩进，而且其上下都有竖直间距；然而随着嵌套层数的增加，这些缩进量是减小的。最多允许嵌套六层。

4.3 列表

要生成有格式的列表，有三种可以使用的环境：

```
\begin{itemize}      列表文本 \end{itemize}
```

```
\begin{enumerate}   列表文本 \end{enumerate}
```

```
\begin{description} 列表文本 \end{description}
```

在这些环境中每一个的列表文本都要相对于左边界进行缩进，并且具有一个标签或记号。标签的类型就与所用的环境有关。生成标签的命令是 `\item`。

4.3.1 itemize 样例

- 每一项前面有一个黑点，即所谓的 *bullet*，做为标签。
- 每一项中的文本可以具有任意长度。标签位于文本的第一行开头处。
- 相邻项之间自动用额外的竖直间距分开。

上述结果是用如下输入生成的：

```
\begin{itemize}
```

```
\item 每一项前面有一个黑点，即所谓的 \emph{bullet}，做为标签。
```

```
\item 每一项中的文本可以具有任意长度。标签位于文本的第一行开头处。
```

`\item` 相邻项之间自动用额外的竖直间距分开。

`\end{itemize}`

4.3.2 enumerate 样例

1. 标签由相邻的数字组成。
2. 每调用一次 `enumerate` 环境，标签都是从 1 开始编号。

上面的例子是由如下的输入文本生成的：

```
\begin{enumerate}
\item 标签由相邻的数字组成。
\item 每调用一次 \texttt{enumerate} 环境，标签都是从 1 开始编号。
\end{enumerate}
```

4.3.3 description 样例

目的 这个环境适用于给出一组单词或表达式的定义。

例子 关键词做为标签，每一项由分类或解释组成。

其他用途 也可以用于参考文献中的作者列表。

上面的样例是如下生成的：

```
\begin{description}
\item[目的] 这个环境适用于给出一组单词或表达式的定义。
\item[例子] 关键词做为标签，每一项由分类或解释组成。
\item[其他用途] 也可以用于参考文献中的作者列表。
\end{description}

\item[选项] 命令中包含一个可省略的参数，它以黑体形式成为标签。
```

4.3.4 嵌套列表

上面给出的列表环境也可以彼此嵌套，深度可达四层。所用标签要视嵌套层数而定。缩进总是相对于包围它的列表左边界来确定的。下面就是一个四重的 `itemize` 环境：

- 第一层的标签是一个黑点，即 `bullet`。
 - 第二层的标签就是一条长破折号。
 - * 第三层的是一个星号。
 - 第四层的标签就是单个点了。
 - 同时，随着嵌套层数的增加，竖直间距在减少。
 - * 返回第三层。
 - 返回第二层。
- 现在我们又回到 `itemize` 环境的第一层了。

`enumerate` 环境与 `itemize` 类似，随着嵌套层数的增加，标签的样式也在改变：

1. 第一层的标签是阿拉伯数字后跟一个小点。
- (a) 在第二层，标签就是位于小括号 () 内的小写字母。

i. 第三层就用小写罗马数字后跟点号来标识。

A. 而在第四层，用的就是大写字母。

B. 可以用后面一节的方法来改变标签样式。

ii. 返回到第三层了。

(b) 返回到第二层了。

2. 现在又回到 `enumerate` 环境的第一层了。

下面是一个混合类型的嵌套例子：

- 第一层的 `itemize` 标签是 `bullet`。

1. 这里的标签是阿拉伯数字，因为这是 `enumerate` 环境的第一层。

- 这是嵌套的第三层，但却是第二层的 `itemize`。

- (a) 这是全部嵌套的第四层，但只是第二层 `enumerate` 环境。

- (b) 因此标签是位于小括号 () 内的小写字母。

- 在这一层的标签是一长破折号。

2. 每个列表都至少应该有两项。

- 在 `\item` 命令前的空行是没有用的。

上面这个混合列表是用如下输入得到的：

```
\begin{itemize}
\item 第一层的 \texttt{itemize} 标签是 bullet。
\begin{enumerate}
\item 这里的标签是阿拉伯数字，因为这是...
\begin{itemize}
\item 这是嵌套的第三层，但却是...
\begin{enumerate}
\item 这是全部嵌套的第四层，但只是...
\item 因此标签是位于小括号 () 内的小写字母。
\end{enumerate}
\item 在这一层的标签是一长破折号。
\end{itemize}
\item 每个列表都至少应该有两项。
\end{enumerate}

\item 在 \item 命令前的...
\end{itemize}
```

4.3.5 改变标签样式

很容易地改变 `itemize` 和 `enumerate` 环境中的标签，方法就是利用 `\item` 的可省参数。利用 `\item[+]`，标签就变为 +，而若使用了 `\item[2.1:]`，标签就成为 2.1:。这里的可省参数会取代标准标签。但是对于 `enumerate` 环境，这就意味着对应的计数器并不能自

动增加, 用户必须手工改变记数器的值。

可省标签在专为标签保留的区域中右对齐。该区域的宽度是该层次的缩进总长度减去标签与正文的间距, 这意味着标签区域的左边界与包围它的外层环境的左边界是对齐的。

可以改变文档中全部或部分的标准标签。在 \LaTeX 中是用如下内部命令给 `itemize` 生成标签的:

```
\labelitemi \labelitemii \labelitemiii \labelitemiv
```

而相应于 `enumerate` 环境的命令则是:

```
\labelenumi \labelenumii \labelenumiii \labelenumiv
```

这里每条命令名称末尾的 `i`, `ii`, `iii` 和 `iv` 分别指的是四个可能的层次。

可以用 `\renewcommand` 来改变这些命令。例如, 要把第三层的 `itemize` 标签从 `*` 改为 `+`, 可以用

```
\renewcommand{\labelitemiii}{+}
```

同样, `enumerate` 环境中的标准标签也可以改变。然而, 这里需要的操作就要复杂一些, 因为每个 `enumerate` 层次都对应着一个记数器, 名字分别为 `enumi`, `enumii`, `enumiii` 和 `enumiv`。可以用命令 `\arabic`, `\roman`, `\Roman`, `\alph` 或 `\Alph` 显示记数器的值, 这些命令的意义从名字上就很容易知道, 即 `\Roman{xyz}` 用大写罗马数字显示出记数器 `xyz` 的当前值, 而 `\alph{xyz}` 则用的是小写字母 (这里 `a` 对应于 1, `z` 对应于 26) 显示记数器的值。

这些记数器同记数器样式命令相结合, 就可以重定义 `enumerate` 的标签。例如, 要把第二层的标签改为阿拉伯数字后加 `'.'`, 那就需要用

```
\renewcommand{\labelenumii}{\arabic{enumii}.}
```

这样就把 `\labelenumii` 重定义为用阿拉伯数字显示的记数器 `enumii` 的值, 后跟 `'.'`。所有的层次的编号都可以用这种方法改变, 而且在重定义中可以同时使用多个记数器:

```
\renewcommand{\labelenumii}{\Alph{enumi}.\arabic{enumii}}
```

这样就会使得每当在第二层 `enumerate` 环境中调用 `\item` 命令时, 所显示出来的标签形式为: 大写字母显示的记数器 `enumi` 的值, 后跟数字形式显示的记数器 `enumii` 的值, 例如: A.1, A.2, ..., B.1, B.2, ..., 等等。

如果要在整个文档中应用新的标准标签, 那就应当把重定义命令放在导言中。否则, 它就只在其所出现的那个环境内有效。

利用这种方法, 可以把 `itemize` 和 `enumerate` 环境中的标签改变为任何所期望的样式。然而, 我们不鼓励进行这种修改, 因为从排版的角度来看, 由 Leslie Lamport 所选定的标准标签是很难再做什么改进的。如果确实需要其他的设置, 那就应当在整个文档中都使用这个设置。

4.3.6 参考文献

科技出版物中通常都会包含一长串参考文献, 它由与文章相关的其他作品的名称组成, 在正文中通过其活动编号对其进行引用。通常正文没有结束, 参考文献就不会完成。

每向参考文献中插入一项，就会造成编号的改变，这时需要通读所有的正文，以改变相应的引用编号，所以这是一件非常令人头痛的事。因此 L^AT_EX 提供了一种新的机制，它不但对参考文献进行格式化排版，而且跟踪对它进行的修改和添加操作，以在正文中自动改变引用。

参考文献可以用如下环境来生成：

```
\begin{thebibliography}{标签样本}
    所有的参考文献项
\end{thebibliography}
```

在参考文献中的每一项都是用如下命令生成的：

```
\bibitem[ 标签 ]{ 关键词 } 文本条目
```

如果没有给出可省参数 标签，\bibitem 就会生成一个位于中括号内的活动编号，以便在正文中引用。如果想给出 标签，那可以用任何内容做标签，例如作者姓名的缩写，或者随意的一个引用编号。不可省参数 关键词 赋给该项一个名称，这个名称可以用在正文中，以标志对该项的引用。引用关键词最终不会出现在参考文献被格式化后的结果中，而且位于正文中的引用关键词最终被标签取代。引用关键词可以由字母、数字和除了逗号外的所有符号组成。

真正的参考文献信息是包含在 文本条目 中的，其形式可能为‘作者，题目，出版社，年代，版本，页码’，而且不同部分字样可能不同。在第一行后面的文本要进行缩进，缩进宽度等于 标签样本 的宽度，因此标签样本应是参考文献中的最长标签。如果采用的是活动编号，标签样本 应该是一个没有意义的数字，但其位数等于最大标签的位数（如果在参考文献中的项数超过 10 个，但不足 100 个，可以用 99 做为标签样本）。

在正文中的引用由如下命令生成：

```
\cite{ 关键词 }
```

这里的 关键词 就是出现在 \bibitem 命令中的引用关键词。例如，在参考文献环境中输入了如下文本：

```
\begin{thebibliography}{99}
    \bibitem{adobe} Adobe Systems, PostScript Language Reference Manual, 2nd
        edition, Addison--Wesley, 1990.
    . . . . .
    \bibitem{lgs} Goosens, M., Rahtz, S., and Mittelbach, F., The
        \LaTeX{} Graphics Companion---Illustrating Documents with \TeX{} and
        PostScript, Addison--Wesley, 1997.
    . . . . .
    \bibitem{knuth} Donald E. Knuth. \textsl{Computers and
        Typesetting Vol. \ A--E}. Addison--Wesley Co., Inc.,
        Reading, MA, 1984--1986
    \bibitem[5a]{knuth:a} Vol A: \textsl{The \TeX book}, 1984
    \bibitem[5b]{knuth:b} Vol B: \textsl{\TeX: The Program.}, 1986
    . . . . .
```

```
\end{thebibliography}
```

那么文本

```
For additional information about \LaTeX\ and \TeX\ see
\cite{lgr} and \cite{knuth, knuth:a}.
```

的结果为: For additional information about \LaTeX and \TeX see [3] and [5,5a].

这里 `lgr`, `knuth` 和 `knuth:a` 被选作关键词。用 99 做标签样本, 因为对于这里的 `\bibitem` 标准形式, 两位数的标签就生成足够的缩进。关键词为 `knuth` 的项是列表中的第五项, 因此自动得到标签 [5]。为了使其子项 `knuth:a, ...` 的编号为 [5a], ..., [5e], 需要把可省的标签参数设置为 5a, ..., 5e。

`thebibliography` 环境的结果是以参考文献的形式列于文档尾部。对于文档类 `book` 和 `report`, 在开头会显示一个单词 **Bibliography** 做为章标题, 而对于 `article` 类, 则会显示 **References** 做为节标题。另外, 在 CCT 中文 $\text{\LaTeX} 2_{\epsilon}$ 中, 会在参考文献的上方显示中文名称“参考文献”。上面的那个参考文献示例结果为:

- [1] Adobe Systems, PostScript Language Reference Manual, 2nd edition, Addison-Wesley, 1990.
-
- [3] Goosens, M., Rahtz, S., and Mittelbach, F., The \LaTeX Graphics Companion — Illustrating Documents with \TeX and PostScript, Addison-Wesley, 1997.
-
- [5] Donald E. Knuth. *Computers and Typesetting Vol. A-E*. Addison-Wesley Co., Inc., Reading, MA, 1984–1986
- [5a] Vol A: *The \TeX book*, 1984
- [5b] Vol B: *\TeX : The Program.*, 1986
-

4.4 广义列表

我们前面介绍了 `itemize`, `enumerate` 和 `description` 三种列表环境, 实际上可以用相当一般的形式构造出这些列表环境。标签的类型与宽度、缩进的宽度、段落和标签之间的距离等等, 都可以由用户通过如下的 `list` 环境来全部或部分地进行设置:

```
\begin{list}{标准标签}{列表声明}列表中的项\end{list}
```

这里的 列表中的项 由列表的各项文本组成, 每一项由 `\item` 命令开头, 它生成相应的标签。当 `\item` 命令没有参数时, 就用 标准标签 所包含的内容生成标准标签 (见后)。

在 4.4.2 节中描述的列表参数可以由用户通过 列表声明 设为任何所期望的值。

4.4.1 标准标签

在 `list` 环境中的第一个参数是 标准标签, 当 `\item` 命令没有参数时, 要用它生成标签。对于无变化的标签, 如 `itemize` 环境中的标签, 只要在这里输入要做为标签的符

号就可以了。如果希望它是一个数学符号，那就必须以 $\$$ 符号名 $\$$ 形式给出，即包含在 $\$$ 符号内。例如，要用 \Rightarrow 做标准标签，标准标签就必须定义为 $\$ \backslash Rightarrow \$$ 。

然而，通常标签中需要包含一系列数字。为此，就必须用 $\backslash newcounter\{名称\}$ 命令生成一个新的计数器，这里的名称就是它的标号。在 list 环境中第一次使用这个计数器之前，就必须给出这条命令。假设已为此而定义了一个计数器 marker，那么标准标签参数就可以用 4.3.5 节中所给出的命令来显示计数器的值：例如， $\backslash arabic\{marker\}$ 就生成一个活动的阿拉伯数字编号。即使再复杂的标签也可以用这种方法得到。如果接连的编号为 A-I, A-II, ..., 标准标签就要设为 $A--\backslash Roman\{marker\}$ 。

如果想在标准标签参数里正常使用某个计数器，那么必须先要在列表声明中包含 $\backslash usecounter\{计数器\}$ 命令，从而把该计数器与列表关联起来，这里的计数器就是赋给计数器的名称（上面例子中的 marker）。

标准标签实际上是由 $\backslash item$ 命令调用 $\backslash makelabel\{标签\}$ 得到的。用户在列表中如果想重定义 $\backslash makelabel$ ，可以借助于 $\backslash renewcommand$ 命令：

$\backslash renewcommand\{\backslash makelabel\}\{新的定义\}$

如果标准标签是以这种方式定义的，在 list 环境中相应的项左边就是这里的新标签。因为 $\backslash makelabel$ 是非常一般的命令，它覆盖了其他定义。在 7.5.9 节中有示例。

4.4.2 列表的样式参数

有很多样式参数可对列表进行格式化， \LaTeX 把它们都设置为特定的标准值。在特定的列表中，用户可以在列表声明中改变这些值。可以像通常那样用 $\backslash setlength$ 命令赋值。然而，如果这种赋值是在 list 环境外进行的，在大多数情形中，就被简单地忽略了。这是因为在每一层每一个参数都被预设置为默认值，只有列表声明才可以覆盖这些默认值。

下面列出了样式参数，同时在图 4.1 中进行了标识，它们是基于 L^AT_EX 的选择：

$\backslash topsep$

是一个竖直距离，它要和 $\backslash parskip$ 一起插入到列表与包围列表的上下文之间。在每一个列表层次中都被重新设置为默认值，所以不能在列表声明外面进行全局重定义。

$\backslash partopsep$

当第一个 $\backslash item$ 项前而有空行，或者最后一个 $\backslash item$ 项后面有空行时，就需要把这个量与 $\backslash topsep + \backslash parskip$ 一起插入到列表的上面和下面。可以进行全局重定义，但只限于第一、二层。

$\backslash parsep$

它是在一个 $\backslash item$ 项中相邻两个段落之间的竖直距离。同 $\backslash topsep$ 一样，也会在每一个层次上被设置成默认值。

$\backslash itemsep$

它是一个竖直距离，和 $\backslash parsep$ 一起插入到两个 $\backslash item$ 项之间。同前面的 $\backslash topsep$ 和 $\backslash parsep$ 一样，其默认值也是在每一个层次上被重新设置，因此不能进行全局

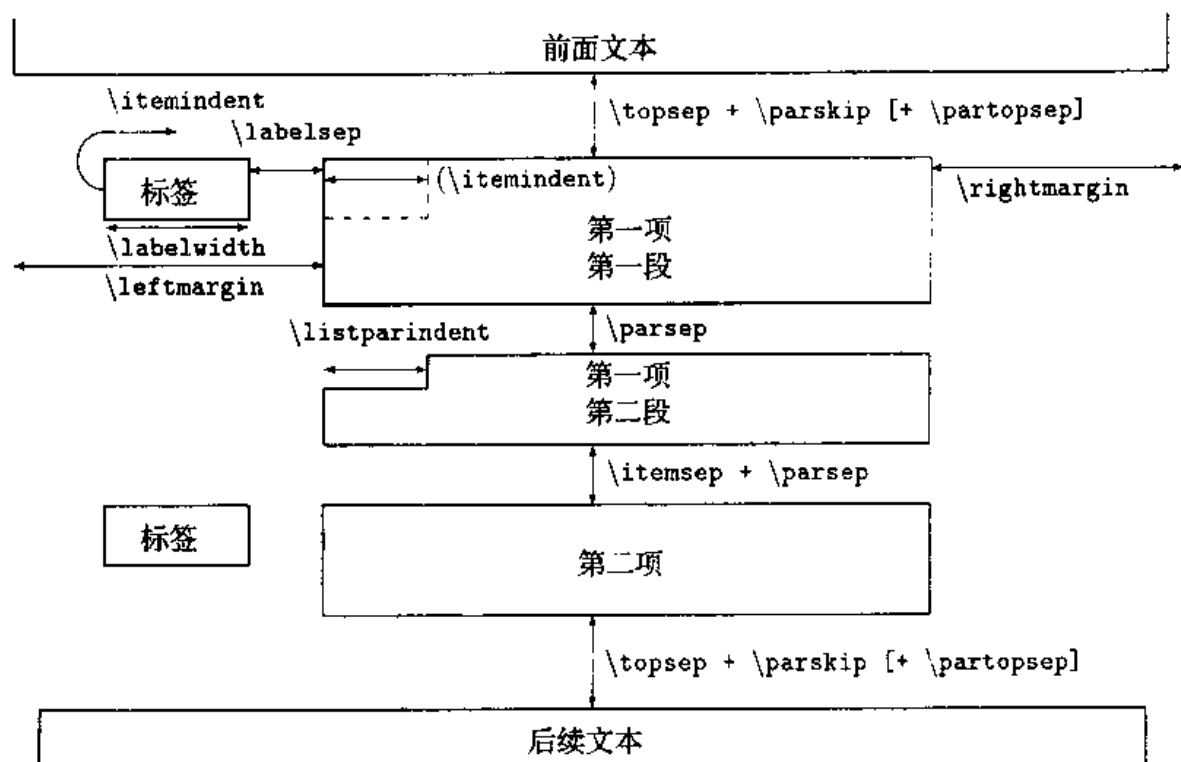


图 4.1 list 参数

性的改变。

`\leftmargin`

是当前环境的左边界到列表文本左边界的距离。其在每个层次上都有默认值，但可以如 4.4.6 节中描述的那样进行全局重定义。

`\rightmargin`

是当前环境右边界到列表文本右边界的距离。其标准值为 0pt，只可以在列表声明中对其进行改变。

`\listparindent`

是每一个 `\item` 项中一个段落的第一行相对于列表文本左边界的缩进宽度。通常设为 0pt，因此没有缩进。只可以在列表声明中改变其值。

`\labelwidth`

是为标签预留的盒子宽度。在这个空间中，标签是右对齐的。可以为其设置一个全局的默认值，以适用于所用的列表层次。

`\labelsep`

是标签盒子与列表文本之间的距离。可以全局性地赋一个新值，但它只对第一层有作用。

`\itemindent`

是一个 `\item` 项中标签及文本第一行向右的缩进宽度。通常取为 0pt，因此就不进行这种缩进。只可以在列表声明中改变其值。

当把竖直距离从其标准值变为其他值的时候，最好使用弹性长度 (2.5.2 节)。

由 `\item` 命令生成的标签通常在一个宽度为 `\labelwidth` 的盒子中是右对齐的。也可以如下面参数列表那样，把它设为左对齐的，即在标准标签的定义或者 `\makelabel` 命令参数的尾部加上 `\hfill`。

4.4.3 用户制作列表的示例

一个关于图的列表为：

Figure 1: *Page format with head, body, and foot, showing the meaning of the various elements involved.*

Figure 2: *Format of a general list showing its elements.*

Figure 3: *A demonstration of some of the possibilities for drawing pictures with \LaTeX .*

这个列表是用如下输入生成的：

```
\newcounter{fig}
\begin{list}{\bfseries\upshape Figure \arabic{fig}:}
  {\usecounter{fig}
  \setlength{\labelwidth}{2cm}\setlength{\leftmargin}{2.6cm}
  \setlength{\labelsep}{0.5cm}\setlength{\rightmargin}{1cm}
  \setlength{\parsep}{0.5ex plus0.2ex minus0.1ex}
  \setlength{\itemsep}{0ex plus0.2ex} \slshape}
  \item Page format with head, body, and foot, showing the
    meaning of the various elements involved.
  \item Format of a general list showing its elements.
  \item A demonstration of some of the possibilities for
    drawing pictures with  $\LaTeX$ .
\end{list}
```

命令 `\newcounter{fig}` 建立了一个新的计数器 `fig`。标准标签设置为直立的黑体单词 **Figure**，后接活动的阿拉伯数字，由 `:` 结尾。每个 `\item` 命令都会显示出这样的标签。

列表声明把 `\usecounter{fig}` 做为其第一条命令，这样就可以在列表中使用 `fig` 计数器。在上面的例子中，标签盒子的宽度 (`\labelwidth`) 被设为 2.0cm，列表文本的左边界 (`\leftmargin`) 为 2.6cm，在标签和文本之间的距离 (`\labelsep`) 为 0.5cm，列表的右边界 (`\rightmargin`) 离包围它的文本距离为 1cm。

在一个 `\item` 项中相邻段落之间的竖直距离 (`\parsep`) 为 0.5ex，但在此基础上还可以伸展 0.2ex，或收缩 0.1ex。项与项之间的额外距离 (`\itemsep`) 为 0ex，可以伸展 0.2ex。

对于所有其他的列表参数，用的都是标准值。在列表声明中最后一条命令是 `\slshape`，它把列表文本设为 *slanted* 字样。

注意: 在 \LaTeX 2_ϵ 中, 如果在标签定义中没有使用 \upshape 声明, 那么每一个 \item 项的标签文本就是 *slanted* 字样的, 例如 **Figure 1:**。在 \LaTeX 2.09 中, 就必须用 \bf 和 \sl 声明, 但是用这种方法不可能得到一个黑体的 *slanted* 字样。

4.4.4 做为新环境的列表

如果要在一个文档中多次使用同一类型的列表, 那么就需要在列表环境中重复输入同样的 标准标签 和 列表声明, 这是一件非常麻烦的事。为此 \LaTeX 提供了一种方法, 可以把给定的列表定义为具有自己名称的环境。这是用 \newenvironment 命令实现的。

例如, 可以用名称 `figlist` 保存上面例子中的列表, 以后利用该名称就可以调用这个列表:

```
\newenvironment{figlist}{\begin{list}
  {\bfseries\upshape Figure \arabic{fig}:}
  {\usecounter{fig} ... {0ex plus0.2ex}\slshape}}
{\end{list}}
```

有了上述定义, 就可以如下调用:

```
\begin{figlist} 列表项 \end{figlist}
```

其使用方式同 \LaTeX 预定义的列表环境相同。

\LaTeX 自身就经常利用列表环境定义其他的一些结构。例如, `quote` 环境的定义为

```
\newenvironment{quote}{\begin{list}{}
  {\setlength{\rightmargin}{\leftmargin}}
  \item[]}{\end{list}}
```

因此该环境也就是一个列表, 其 \rightmargin 的值被设置为 \leftmargin 的当前值, 它们的默认值为 2.5em. 列表本身只由一个 \item 项组成, 它没有标签, 在 `quote` 的定义中自动调用了 \item[] 以达此目的。

采用同样的方式, \LaTeX 在内部通过特殊的 `list` 环境定义了 `quotation` 和 `verse` 环境。左边界以及与包围文本的竖直距离都取的是 `list` 环境的标准值, 因此只有当标准值改变时, 这些距离才会变化。

最后, 我们给出一个自定义特殊列表的例子:

```
\newenvironment{lquote}{\begin{list}{}{\item[]}{\end{list}}
```

这生成一个 `lquote` 环境, 它使所包含的文本相对于外部文本进行了 \leftmargin 的缩进, 右边界与外部文本对齐 (因为 \rightmargin 的标准值为 0pt), 除此之外并没有其他的修改。

4.4.5 平凡列表

在 \LaTeX 中还包含一个 `trivlist` 环境, 其语法为

```
\begin{trivlist} 内部文本 \end{trivlist}
```

在这里没有 标准标签 和 列表声明 参数。它相当于一个列表, 其标签是空的, \leftmargin ,

`\labelwidth` 和 `\itemindent` 都被赋值为 0pt, 而 `\listparindent` 的值与 `\parindent` 的值相同, `\parsep` 的值与 `\parskip` 的值相同。

L^AT_EX 利用 `trivlist` 环境生成其他环境。例如, 当调用 `center` 环境时, 在 L^AT_EX 内部实际上调用的就是

```
\begin{trivlist} \centering \item[] 内部文本 \end{trivlist}
```

环境 `flushleft` 和 `flushright` 的定义方式与 `center` 类似。

4.4.6 嵌套列表

广义列表和平凡列表也可以与其他的列表环境 `itemize`, `enumerate` 和 `description` 相互嵌套, 最大嵌套深度为 6 层。在每一层, 都要相对于前面高一级那层进行总量为 `\leftmargin` 的缩进。

前面已提到过, 在导言中利用声明只能改变有限的几个 `list` 参数的标准值。对于不同嵌套层数中左边界的缩进量, 虽然每层的标准值是不同的, 我们也可以在导言中改变它们的值。在 L^AT_EX 内部, 这些缩进量是用 `\leftmarginn` 参数来设置的, 这里的 *n* 代表 i, ii, iii, iv, v 或 vi。我们可以改变这些值, 例如用声明 `\setlength{\leftmarginiv}{12mm}` 可以使得第四层的左边界相对于第三层右移 12mm。但要注意, 这些声明必须放在 `list` 环境外面, 而不能放在列表声明里。

嵌套列表的每一层都要调用内部宏 `\@listn` (*n* 从 i 到 vi)。这条命令使 `\leftmargin` 的值等于相应的 `\leftmarginn` 的值, 但是如果在 `list` 环境中显式地声明了 `\leftmargin` 的值, 那就要采用新值。也就是说, 在外部并不存在单个的 `\leftmargin` 标准值, 而是 6 个不同的值。只有在 `list` 环境内部, 参数 `\leftmargin` 才有意义。

4.5 定理型的声明

在科技文献中, 经常需要如下结构:

Theorem 1 (Hilbert) *Every ideal I in $k[x_1, \dots, x_n]$ has a finite generating set. In other words, given an ideal I , there exists a finite collection of polynomials $\{f_1, \dots, f_s\} \subset k[x_1, \dots, x_n]$ such that $I = \langle f_1, \dots, f_s \rangle$.*

或者

Axiom 4.1 *The natural numbers form a set S of distinct elements. For any two elements a, b , they are either identical, $a = b$, or different from one another, $a \neq b$.*

对上述的结构, 除了采用这里的名称 Theorem 或 Axiom 外, 有时还会有名称 Definition, Corollary, Declaration 或 Lemma。它们的共同特点就是都有以黑体字样显示的关键词和活动编号, 而其他的内容则是斜体显示。

当然, 用户可以通过显式给出样式和编号, 从而生成上述结果, 但是, 如果需要在中间插入同样类型的一个新结构, 那么用户就必须对后面的结构进行重新编号。不过, 这样操作是很麻烦的。为此, 我们可以利用命令

```
\newtheorem{结构类型}[结构标题][其他计数器]
```

生成同样的结果, 这时 L^AT_EX 会自动跟踪编号的变化。这里的 结构类型 是用户给该结构指定的一个名称, 而 结构标题 就是那个以黑体字样与活动编号列在一起的关键词 (如 Theorem)。如果没有可省参数 其他计数器, 那么在整篇文档中该结构是连续编号的。然而, 如果在 其他计数器 中有一个已定义了的计数器的名称, 如 chapter, 每当该计数器增加时, 后接结构的编号就又从 1 开始, 而且我们也可以把多个计数器列在一起, 例如上面的 Axiom 4.1。

预定义的结构可以用如下命令来调用:

```
\begin{结构类型}[附加标题] 文本 \end{结构类型}
```

这里会给相应的计数器增 1, 以生成恰当的编号。上面的例子就是用如下输入得到的:

```
\newtheorem{theorem}{Theorem} \newtheorem{axiom}{Axiom}[chapter]
. . . . .
\begin{theorem}[Hilbert] Every ideal ..... \end{theorem}
\begin{axiom} The natural numbers form ..... \end{theorem}
```

可以省略的 附加标题 紧接在活动编号后以黑体字样显示在小括号 () 内。

有时候某个结构需要与其他结构一起编号, 而没有自己的计数器。为此可以在定义中包含另一个可省参数:

```
\newtheorem{结构类型}[编号来源]{结构名称}
```

这里的 编号来源 是一个已存在的定理结构名称, 它们共享同一个计数器。因此通过定义 \newtheorem{subthrm}[theorem]{Sub-Theorem}, theorem 和 subthrm 这两个结构的编号是同一个序列: Theorem 1, Sub-Theorem 2, Sub-Theorem 3, Theorem 4, 依次类推。

4.6 制表位

4.6.1 基础知识

在打字机上可以在一行的任何位置设置制表位; 这样当按一些 tab 键, 打字头或者回车键就会跳到下一个制表位处。

在 L^AT_EX 的 tabbing 环境中也可以实现类似的功能:

```
\begin{tabbing} 文本行 \end{tabbing}
```

可以认为, tab 停留点按从左到右具有顺序编号。在 tabbing 环境开始, 除了左边界 (它称为第零个制表位) 外, 没有其他的停留点。在一行中可以用命令 \= 在任何地方设置制表位, 而一行是用 \\ 命令结束的:

```
Here is the \=first tab stop, followed by \=the second\\
```

这里把第一个制表位设在紧接单词 the 的空白后面, 而第二个是在单词 by 的后面。

当以这种方法设置好了制表位后, 就可以在后续的文本行中从左边界开始, 用命令 \> 跳到任一个制表位处。新行还是用通常的 \\ 命令开始的。

例如:

Type	Quality	Color	Price
Paper	med.	white	low
Leather	good	brown	high
Card	bad	gray	med.

```
\begin{tabbing}
Type\quad\quad= Quality\quad\quad=
Color\quad\quad= Price\quad\quad[0.8ex]
Paper \> med. \> white \> low \>
Leather \> good \> brown \> high\>
Card \> bad \> gray \> med.
\end{tabbing}
```

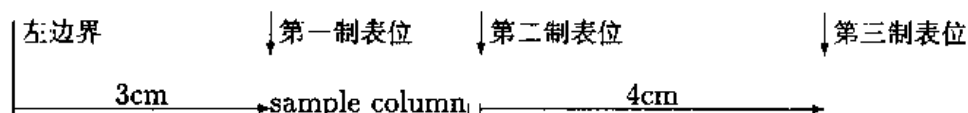
4.6.2 样本行

利用不必显示出来的样本行设置制表位, 是一种比较好的方法, 而且有时也是必须的方法。例如, 样本行可以由后面出现的各列中最宽项组成, 或者由制表位之间最小的列间距组成。在样本行中也可以包含 `\hspace` 命令, 这样可以保证制表位之间的距离为预先确定的长度。

为了不显示样本行, 那么该行文本就不能用 `\>` 结束, 而要用 `\kill` 结束。

```
\hspace*{3cm}\=sample column \=\hspace{4cm}\= \kill
```

除左边界外, 上例共设置了三个制表位:



在样本行开头的 `\hspace` 命令必须是 `*`-形式, 否则就会去掉在行边界所插入的间距。

4.6.3 制表位与左边界

在 `tabbing` 环境中, 每行的左边界首先要与包围该环境的文本左边界一致, 并标记为第零个制表位。通过在一行的开头加上制表键 `\>`, 可以使文本在第一个制表位处开始。然而, 如果在第一行开头处加上 `\=` 命令, 就会使得后续各行都从第一个制表位处开始。用 `\+ \+` 开始或结束一行, 会使得后面的行都是开始于两个制表位后。一行中可以用多少个制表位, 就可以使用多少个 `\+` 命令。

而 `\-` 命令具有与 `\+` 相反的效果, 它把后续各行的左边界向左移动一个制表位。但用这条命令不能把边界设在第零个制表位的左边。

在某一行中要覆盖 `\+` 命令的效果, 可以采用在将要被去掉的制表位前面加上 `\<` 命令的方法, 这样该行的制表位就从当前的左边界开始。对于接在 `\>` 命令的新行, 就在由前面所有的 `\+` 和 `\-` 命令总数所确定的制表位处开始。

4.6.4 其他的制表命令

在任一行上都可以重设或增加制表位。用 `\=` 命令可以重设下一个制表位, 然而如果已经用了足够多的 `\>` 命令, 从而到达了最后一个制表位, 这时用 `\=` 命令就会插入一个制表位。例如:

Old column 1	Old column 2		<code>\begin{tabbing}</code>
			<code>Old column 1 \= Old column 2\\</code>
Left column	Middle col	Extra col	<code>Left column \> Middle col</code>
			<code>\= Extra col\\</code>
New col 1	New col 2	Old col 3	<code>New col 1 \= New col 2 \></code>
			<code>Old col 3\\</code>
Column 1	Column 2	Column 3	<code>Column 1\> Column 2 \> Column 3</code>
			<code>\end{tabbing}</code>

有时候，在重设了制表位后又希望使用原来的制表位。为此，可以利用 `\pushtabs` 命令来实现这一目标，这条命令把当前制表位保存起来，并把它们从当前行上去掉。这样所有的制表位都可以重新设置。可以用 `\poptabs` 命令来激活保存的制表位。可以根据需要多次使用 `\pushtabs` 命令，但在同一个 `tabbing` 环境中必须包含与它数目相同的 `\poptabs` 命令。

可以用 左边文本 \’ 右边文本 在一个制表位处定位文本，这里的 左边文本 指的是恰好位于当前制表位前面（或者左边界）且具有很小距离的文本；而 右边文本 恰好在制表位处开始。在 左边文本 与制表位之间的距离由制表参数 `\tabbingsep` 确定。用户可以如通常那样用命令 `\setlength` 来修改它的值。

可以用 `\’` 文本 命令使文本相对于右边界对齐。但此时必须保证在这一行的余下部分中没有 `\>` 或 `\=` 命令。

`\=`、`\’` 和 `\’` 命令在 `tabbing` 环境外是重音命令（2.6.7 节）。如果在 `tabbing` 环境内需要这些重音，那就必须用 `\a=`、`\a’` 和 `\a’` 来代替。例如，要在 `tabbing` 环境内生成 `ó`、`ò` 或 `õ`，那就必须输入 `\a’o`、`\a’o` 或 `\a=o`。`\-` 命令在 `tabbing` 环境外也具有另外一种意义（单词的建议断点），但由于在 `tabbing` 环境中不会自动断行，因此不需要有替代形式。

下面是一个演示所有制表命令的例子：

Apples:	consumed by: people	<code>\begin{tabbing}</code>
	horses	<code>Grapefruits: \= \kill</code>
	and sheep	<code>Apples: \> consumed by: \= people\+\\</code>
	reasonably juicy	<code>horses \\</code>
Grapefruits: a delicacy		<code>and \’ sheep\-\</code>
(see also: melons		<code>reasonably juicy\-\</code>
pumpkins)		<code>Grapefruits: \> a delicacy\\</code>
Horses	feed on	<code>\pushtabs</code>
	apples	<code>(see also: \= melons\\</code>
		<code>\> pumpkins)\\</code>
		<code>\poptabs</code>
		<code>Horses \> feed on \> apples</code>
		<code>\end{tabbing}</code>

4.6.5 关于制表的注解

\TeX 处理 `tabbing` 环境的方式与通常处理段落的方式相同，在环境中如果有必要的话，会进行分页。然而，不允许在其中使用 `\newpage` 和 `\clearpage` 命令，而忽略 `\pagebreak` 命令。如果用户要强迫在 `tabbing` 环境中分页，那么可以使用一个小技巧：在希望分页的那行末尾定义一个相当大的行间距（如 `\\[10cm]`）。这就会强迫在此处分页，而且在下一页上不会出现所定义的空白。

文本行实际上是位于一对 `{ }` 中，因此在某一行上的尺寸或字体声明只对该行有作用。文本不需显式地放在一对大括号内。但要注意一个 `tabbing` 环境中不能嵌套另一个 `tabbing` 环境。

注意：制表移动命令 `\>` 总是移到下一个逻辑制表位处。如果前面的文本长度超过两个制表位之间可用的间距时，这实际上是一个回移。这与打字机上的制表机制是不同的。

在 `tabbing` 环境中不会自动进行断行。在没有遇到 `\\` 命令之前，一行是不会断开的。这样文本就有可能延伸超过右边界，从而需要用户来决定到底该如何处理。

在 `tabbing` 环境中 `\hfill`、`\hrulefill` 和 `\dotfill` 命令是没有作用的，因为这里不会发生伸展。

4.7 盒子

所谓盒子，就是一部分文本，被 \TeX 当做整体对待，如同单个字符那样进行处理。一个盒子（连同其中的文本）可以上下左右移动。由于盒子是一个整体，即使最初它是由更小的不可分的盒子组成， \TeX 也不能再把它分开。然而，如果乐意的话，可以把更小的盒子放在一起，以构造一个更大的盒子。

这实际上也就是 \TeX 内部进行格式化排版的方法：单个字符构成字符盒子，然后把它放到水平的行盒子中，在单词间插入弹性长度。行盒子竖直堆积，构成段落盒子，行间要插入弹性长度。然后把它们放到页面主体盒子中，其连同页眉和页脚盒子一起构成页面盒子。

在 \LaTeX 中，用户可以选择三种类型的盒子：LR 盒子，段落盒子和标尺盒子。LR（左-右）盒子由水平的从左到右的有序材料组成。段落盒子则由竖直堆积的行组成。标尺盒子是一个用黑色填充的实心矩形，通常用来画水平线或竖直线。

4.7.1 LR 盒子

要创建一个由处于 LR 模式的文本组成的盒子，可以用命令

```
\mbox{ 文本 }      和      \makebox[ 宽度 ][ 位置 ]{ 文本 }
\fbbox{ 文本 }     和      \framebox[ 宽度 ][ 位置 ]{ 文本 }
```

左边两条命令生成一个 LR 盒子，其宽度恰好是在 `{ }` 中所给文本的宽度。`\fbbox` 命令同 `\mbox` 命令一样，只是文本要用方框包围起来。

而对于右边两条命令，宽度是由可省的长度参数 `宽度` 来预先确定的。另一个可省参数 `位置` 用来定义文本在盒子的位置。如果不给任何值，文本是居中排列的。位置参

数可以取如下值:

- l 文本左对齐,
- r 文本右对齐,
- s 伸展文本, 以达到所定义的宽度。

注意, s 是 L^AT_EX 2_ε 新增加的类型。因此 `\makebox[3.5cm]{centered text}` 生成一个宽度为 3.5cm 的盒子, 其中的文本是居中排列的, 即 `centered text`, 多余地方是用空白填充的, 而 `\framebox[3.5cm][r]{right justified}` 的结果是文本在一个宽度为 3.5cm 的盒子中右对齐:

right justified

。对于 L^AT_EX 2_ε, 也可以用

`\framebox[3.5cm][s]{stretched\dotfill text}`

以生成一个填满的盒子:

stretched.....text

, 在这种情形中需要利用弹性长度或者其他的填充物 (49 页) 以填充由于伸展而出现的空白。

如果文本的自然宽度大于所定义的宽度, 那就会视位置参数的不同, 而向盒子的左边、右边或两边突出。例如,

`\framebox[2mm]{centered}` 结果为

centered

上面这种用法对于 `\framebox` 来说, 确实显得有点儿愚笨, 但是对于 `\makebox` 却可能非常有用。在 `picture` 环境的图示中, 可以利用 `\makebox` 定义一个宽度为 0pt 的盒子, 这样就可以生成一个居中的, 或左(右)对齐的文本 (见第六章的例子)。用它也可以使得两部分文本重叠, 例如 `\makebox[0pt][l]{/}S` 的结果为一条斜线穿过 S, 即 \$。

注意: 长度的定义中必须包含长度单位, 即使所定义的长度是零。因此这里定义零宽度必须用 0pt, 而不能只用 0。

在 L^AT_EX 2_ε 中也可以利用 LR 盒子的自然宽度 (即只用命令 `\mbox` 时的宽度) 来定义其宽度:

<code>\width</code>	表示盒子的自然宽度,
<code>\height</code>	表示从基线到顶部的距离,
<code>\depth</code>	表示从基线到底部的距离,
<code>\totalheight</code>	就是 <code>\height</code> 加上 <code>\depth</code> 。

要生成一个宽度为其中包含文本的总高度六倍的有框盒子, 可以用命令:

`\framebox[6\totalheight]{Text}`

Text

注意: 这些特殊的长度参数只在一个 LR 盒子的宽度定义或者下面将要介绍的段落盒子的高度定义中才有意义。而出现在其他情形中, 就会出现错误信息。

如果一部分文本要在文档的几个地方出现, 那么可以把它们保存起来, 首先采用如下命令:

`\newsavebox{\盒子名}`

这样就可以创建一个盒子, 其名称为 '`\盒子名`'。该名称要符合 L^AT_EX 命令名称的语法 (只能有字母), 并且前面有 `\`。它也不能与任何已存在的 L^AT_EX 命令同名。当用上面的命令初始化了一个盒子后, 那么命令:

`\sbox{\盒子名}{文本}` 或者
`\savebox{\盒子名}[宽度][位置]{文本}`

就把文本的内容保存起来,以备后面使用。这里的可省略参数 `宽度` 和 `位置` 与 `\makebox` 和 `\framebox` 中的意义一样。接着就可以在需要时用命令

```
\usebox{\盒子名}
```

把保存的文本当做一个整体插入到文档中。

LR 盒子的内容也可以用如下环境来保存:

```
\begin{lrbox}{盒子名}
  文本
\end{lrbox}
```

其等价于 `\sbox{\盒子名}{文本}`。使用这种环境的好处在于它可以保存用户定义的环境 (7.4 节) 中的文本, 以备将来用 `\usebox` 插入。

4.7.2 LR 盒子的竖直移位

命令

```
\raisebox{上移量}[高度][深度]{文本}
```

生成一个内容为文本的 `\mbox`, 它相对于当前基线向上移动了上移量。可省参数告诉 \LaTeX 该盒子在基线上方伸展的高度, 基线以下的深度。若没有使用这些参数值, 那么盒子的尺寸就是由文本和上移量确定的自然尺寸。注意这里的上移量、高度和深度都是长度 (2.5.1 节)。如果上移量为负数, 那么盒子就相对于基线向下移动。

例如:

```
Baseline \raisebox{1ex}{high} and \raisebox{-1ex}{low}
and back again
```

结果为: Baseline ^{high} and _{low} and back again.

高度和深度的值可以与文本的实际值完全不同。其结果是根据同一行上所有其他盒子 (字符也是盒子) 的高度和深度, 来确定当前文本行与前后文本行的距离。如果提升了一个盒子, 而高度参数却仍然用的是正常的字符大小, 那么被提升的盒子就会与上面的文本行重叠, 同样当降低一个盒子时, 深度也具有同样的效果。

4.7.3 子段盒子和小页

整个段落可以用如下命令来放到单独的竖直盒子 (或者套用 \LaTeX 术语称为子段盒子) 中:

```
\parbox[位置]{宽度}{文本}
```

也可以用环境实现同样的效果:

```
\begin{minipage}[位置]{宽度} 文本 \end{minipage}
```

它们都生成具有给定宽度的竖直盒子, 其中的文本行如通常段落模式一样彼此堆积。

可省的参数 `位置` 可取如下值:

- b 盒子的底边与当前基线对齐,
- t 顶行文本与当前基线对齐。

当没有任何 `位置` 参数时, 子段盒子相对于外部文本的基线竖直居中排列。

位置参数只有当 `\parbox` 命令或者 `minipage` 环境位于一个段落内部时才有意义, 否则当前行与基线就没有什么意义。如果子段盒子前面紧接一个空行, 这就标志着要开始一个新的段落。在这种情形中, 子段盒子的竖直位置是由段落中后面的元素确定的。而后面的元素也可以仍为子段盒子。如果整个段落只是由一个子段盒子或者小页组成, 那么位置参数就没有意义, 是无效的。例如:

```
\parbox{4.5cm}{\sloppy This is a 4.5 cm wide parbox. It is
    vertically centered on the}
\hfill CURRENT LINE. \hfill
\parbox{6cm}{Narrow pages are hard to format. They usually
produce many warning messages on the terminal. The command
{\tt\symbol{92}sloppy} can stop this.}
```

This is a 3.5 cm wide parbox.
It is vertically centered on the

CURRENT LINE.

Narrow pages are hard to format. They
usually produce many warning mes-
sages on the terminal. The command
`\sloppy` can stop this.

```
\begin{minipage}[b]{4.7cm}
    The minipage environment creates a vertical box like the parbox
    command. The bottom line of this minipage is aligned with the
\end{minipage}\hfill
\parbox{3.5cm}{middle of this narrow parbox, which in turn is
aligned with}
\hfill
\begin{minipage}[t]{5cm}
    the top line of the right hand minipage. It is recommended that
    the user experiment with the positioning arguments to get used
    to their effects.
\end{minipage}
```

The minipage environment cre-
ates a vertical box like the par-
box command. The bottom

line of this minipage is aligned
with the

middle of this narrow
parbox, which in turn
is aligned with

the top line of the right hand
minipage. It is recommended
that the user experiment with
the positioning arguments to get
used to their effects.

在 4.7.4 节中我们将演示如何按所希望的那样把子段盒子彼此竖直堆积起来。

`\parbox` 命令生成一个由文本组成的竖直盒子, 这同 `minipage` 环境是一样的。但

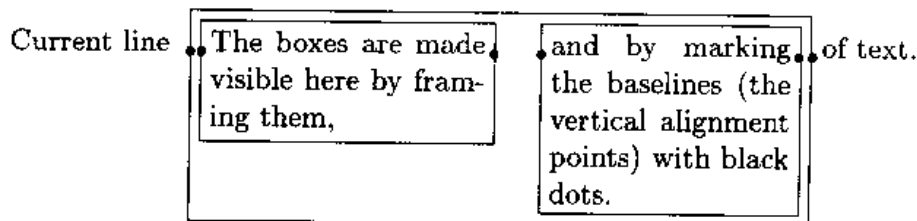
是后者更具有一般性。在 `\parbox` 中的文本不可以包含在 4.2–4.5 节中所讲解的居中、列表或者其他环境。而另一方面，在 `minipage` 环境中是完全可以包含这些环境的。也就是说，一个 `minipage` 可以包含居中或缩进文本，也可以有 `list` 和 `tabbing` 环境。

4.7.4 竖直摆放的问题

对小页或者子段盒子的竖直定位有时会产生意想不到的结果，我们可以用图形来形象地说明 \LaTeX 是如何处理盒子的定位。假设我们想把两个高度不同的盒子并排摆放，并且对齐第一行，而它们的底部与当前文本行对齐。“最明显的”做法就是

```
\begin{minipage}[b]{...}
  \parbox[t]{...}{...} \hfill \parbox[t]{...}{...}
\end{minipage}
```

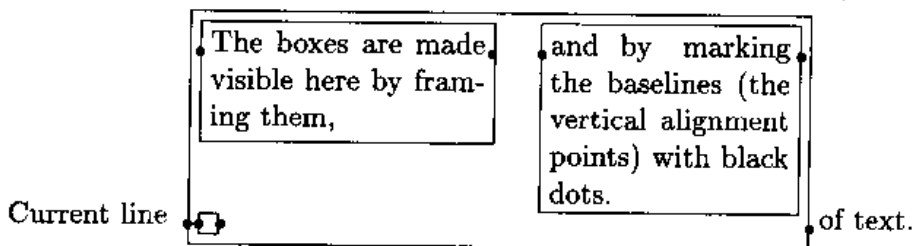
但这行不通，因为它的结果是：



这是因为在 \LaTeX 内部把每个子段盒子或小页环境都当做一个特殊的字符，这些字符也具有自己的相对子基线的高度和深度。当要考虑最外层的小页环境时，它们只是在同一行上的两个“字符”，而且该行同时是顶行和底行。因此最外面的小页的底行确实与文本行对齐，但它同时也是顶行，所以才会显示出上面那样的形状。解决的办法是在最外层盒子中加一个没有内容的第二行：

```
\parbox[t]{...}{...} \hfill \parbox[t]{...}{...} \\ \mbox{}
```

没有内容的行当然不能是真的什么东西也没有，因为这里用了 `\mbox`。



如果要对齐两个盒子的底行，而顶部与当前行对齐，也会有同样的问题出现。下面是两种加入没有内容的第一行的方法：

```
\mbox{} \\ 恰好顶部对齐，或者
\mbox{} \\[-\baselineskip] 第一行文本对齐。
```

在 158 页上有第一种情形的一个例子。在第 83 页上的例子中也用到了空行的技巧。

4.7.5 具有指定高度的段落盒子

实际上，我们前面所讲的段落盒子语法，是适用于 $\text{\LaTeX}2.09$ 和 $\text{\LaTeX}2_{\epsilon}$ 的。在 $\text{\LaTeX}2_{\epsilon}$ 中，对 `\parbox` 命令和 `minipage` 环境的语法已进行了推广，除前面的参数外，还可以有两个可省参数：

```

\parbox[位置][高度][内部位置]{宽度}{文本}
\begin{minipage}[位置][高度][内部位置]{宽度}
  文本
\end{minipage}

```

在这两种情形中,高度定义了盒子的高度;在高度参数中,可以同`\makebox`和`\framebox`(78页)中的宽度参数一样,使用`\height`,`\width`,`\depth`和`\totalheight`参数。

可省参数内部位置说明文本在内部是如何定位的,它只有给定了高度时才有意义。可以取的值为

- t 使文本靠盒子的顶部,
- b 把文本推向盒子的底部,
- c 竖直居中,
- s 伸展文本以填满整个盒子。

在最后那种情形中,当出现竖直伸展时,需要使用弹性长度(2.5.2节)。

注意外部定位参数位置与内部定位参数内部位置的区别:前者说的是盒子如何与包围它的文本对齐,而后者则决定在盒子中的内容是如何安排的。

例:

```

\begin{minipage}[t][2cm][t]{3cm}
  This is a minipage of height 2~cm with the text
  at the top.
\end{minipage}\hrulefill
\parbox[t][2cm][c]{3cm}{In this parbox, the text
  is centered on the same height.}\hrulefill
\begin{minipage}[t][2cm][b]{3cm}
  In this third paragraph box, the text is at the bottom.
\end{minipage}

```

This is a minipage_____


of height 2 cm with the text at the top.	In this parbox, the text is centered on the same height.	In this third para- graph box, the text is at the bottom.
---	--	---

盒子间的`\hrulefill`命令显示了基线的位置。这三个盒子的尺寸是一样的,区别只是内部位置的值不一样。

4.7.6 标尺盒子

所谓标尺盒子就是完全用黑色填充的矩形。其相应命令的一般语法为

```
\rule[提升]{宽度}{高度}
```

这条命令生成一个具有给定宽度和高度的实心盒子,并且相对于基线向上做提升。因此`\rule{8mm}{3mm}`的结果是。若没有可省参数提升,矩形就放在当前文本行的基线上。

参数提升、宽度和高度都是长度(2.5.1 节)。如果提升为负数,则矩形位于基线之下。

也可以生成一个宽度为零的标尺,这样就得到一个不可见的具有给定高度的盒子,这样的构造称为一个支撑,利用它可以强迫其所在的水平盒子具有不同于其所包含内容需要的高度和深度。对于这种情形,用 `\vspace` 就行不通,因为它只是增加已经存在的竖直间距。

例如, `\fbox{Text}` 的结果为 Text。为了得到 Text, 那就必须告诉 TeX, 盒子中的内容要相对于基线向上和向下延伸一定的长度。这可以用 `\fbox{\rule[-2mm]{0cm}{6mm}Text}` 来实现。这也就是说要用方框包围起来的文本是‘一个不可见的柱子,其底位于基线下 2mm, 高有 6mm, 它后面接单词 Text’。我们虽然看不到这个竖直标尺,但它却可以确定方框的顶边和底边。

标尺盒子的高度也可以是零,这就是一个不可见的具有给定宽度的水平线;然而,这种构造却是没有什么实际用途的,因为利用 `\hspace` 命令很容易得到水平位移。

4.7.7 嵌套盒子

上面描述的盒子可以相互嵌套任意次。要把一个 LR 盒子包含在一个子段盒子或者小页中没有任何概念上的困难。相反,也可以把一个子段盒子放在 LR 盒子中,只要头脑中具有这样的想法:每个盒子都是一个整体,TeX 把它们当做具有相应尺寸的单个字符来处理,那么一切就很清楚了。

把子段盒子放在 `\fbox` 命令内部,可以使得整个子段盒子都用方框包围起来。当前的结构就是用

```
\fbox{\fbox{\parbox{12cm}{把子段盒子...}}}
```

得到的一个在方框盒子内的宽度 12cm 的子段盒子,而这个方框盒子又位于另一个方框盒子的内部,这样可以得到双层方框的效果。

把子段盒子包围在 `\raisebox` 内,就可以得到任意所期望的竖直位移。这里的两个盒子都具有位置参数 [b], 而右边的那个是如下生成的:

```
\raisebox{0.5cm}{\begin{minipage}[b]{2.5cm}
  a b c d e ... x y z\\
  \underline{baseline}
\end{minipage} }
```

```
a b c d e f g h i
j k l m n o p q r
s t u v w x y z
baseline
```

它向上位移了 0.5cm。

baseline

在一个 minipage 环境中,两个 minipage 环境相互定位,这是一种很有用的结构。外层 minipage 的位置参数可以把其内容做为一个整体,与相邻文本或者盒子对齐。下面是这种结构的一个示例。

The first line of this 4cm wide minipage or parbox is aligned with the first line of the neighboring minipage or parbox.

This 5cm wide minipage or parbox is positioned so that its top line is at the same level as that of the box on the left, while its bottom line is even with that of the box on the right. The naive notion that this arrangement may be achieved with the positioning arguments set to t, c, and b is incorrect. Why? What should this selection really produce?

The true solution involves the nesting of two of the three structures in an enclosing minipage, which is then separately aligned with the third one.

输入文本为:

```
\begin{minipage}{\textwidth}
\footnotesize
\begin{minipage}[b]{9.5cm}
\begin{minipage}[t]{4cm}
    The first line of this ... ..
\end{minipage}\hfill
\begin{minipage}[t]{5cm}
    This 5cm wide minipage or parbox ... ..
\end{minipage}
\\ \mbox{}
\end{minipage} \hfill
\begin{minipage}[b]{4cm}
    The true solution involves ... ..
\end{minipage}
\end{minipage}
```

由于有可能是左边和中间的结构首先包围在小页中,也可能是中间与右边的结构包围在小页中,因此为了得到上述结果,可以有两种不同的方法。读者可以尝试给出另一种做法。

这里又出现了两个盒子并排放在一起,如何与同行文本对齐的问题(4.7.4节)。为了得到正确地竖直对齐,需要加入一个没有内容的行。

最后提一下,类似于 `\parbox` 命令和 `minipage` 环境等的竖直盒子也可以做为 `\sbox` 或者 `\savebox` 命令中的文本,从而保存起来,以后可以通过 `\usebox` 调用它们,具体步骤请见 4.7.1 节中的描述。

4.7.8 盒子样式参数

对于有框盒子 `\fbox` 和 `\framebox`,它们有两个样式参数,用户可以重设它们的值:

`\fboxrule` 确定方框线的粗细,

`\fboxsep` 设置方框与被包围文本之间的距离大小。

与通常 \LaTeX 的长度参数一样, 可以用 `\setlength` 命令给这些参数赋新值。要想使后面所有 `\framebox` 和 `\fbox` 命令中的线粗都为 0.5mm, 调用 `\setlength{\fboxrule}{0.5mm}` 就可以了。

这些设置的适用范围也遵从通常的规则: 如果其位于导言中, 那么就适用于整个文档; 如果它位于一个环境中, 那么有效性持续到环境结束。

这些参数对于用在 `picture` 环境 (6.4.2 节) 中的 `\framebox` 命令没有作用。因为那里的 `\framebox` 命令的语法和作用要比通常时候的更加广泛。

4.8 显示源文本

有时我们会希望某些文本直接按输入的样子显示, 即不进行格式化处理。这可以用如下命令来实现:

```
\begin{verbatim}      源文本      \end{verbatim}
\begin{verbatim*}    源文本      \end{verbatim*}
```

这样 源文本 就会同输入时的一样, 用打字机字样显示出来, 同时包含所有的空格和断行。通常情况下被当做命令的特殊字符也做为字符串直接显示出来。唯一的例外就是标志着该环境结束的命令 `\end{verbatim}` 本身。这里的显示是从新的一行开始, 当环境结束时, 又从新一行上继续处理 `verbatim` 环境后的正常文本。

这个环境的标准形式与 `*`- 形式的差别在于, 对后一种情形, 空格是用 `\` 符号表示的, 以使得空格更醒目。

例如, 在 89 页上就有一些源文本被直接显示出来, 以演示在禁止模式中脚注的应用。这可以用如下方法得到:

```
\begin{verbatim}
\addtocounter{footnote}{-1}\footnotetext{Small insect}
\stepcounter{footnote}\footnotetext{Large mammals}
\end{verbatim}
```

源文本也可以用如下命令在一行里显示出来:

```
\verb 源文本 c
\verb*c 源文本 c
```

这里的 `c` 为不出现在于 源文本 中的任一字符。它是用来终止这种原样显示模式, 返回通常的文本格式化模式。在 `\verb` 或者 `\verb*` 命令与 `c` 之间不能有空格, 显然, 对于普通形式, `c` 不能是 `*`。

要显示普通文本中的命令名称, 可以用下面这种方法。例如, 为了得到 `'\xyz{ }'`, 输入应该是 `\verb=\xyz{ }=`。这里的等号 `=` 就用来做为定界符 `c`。`*`- 形式的作用类似, 只是同 `verbatim*` 环境一样, 要把空格显示为 `\`: `\xyz{\}`。

标准 \LaTeX 2_ϵ 的宏包 `shortvrb` (第 205 页) 提供了 `\verb` 命令的一种简单紧凑而方便的版本。详情见宏包的使用手册。

同 `verbatim` 环境相比, `\verb` 命令的源文本中不能有断行。如果其中的文本长度超过一行, 断行点就会被当做空白对待 (L^AT_EX 2.09) 或者给出一个错误信息 (L^AT_EX 2_ε)。这个错误信息就会提示你可能忘记输入结束的 `c` 符号。

切记: 无论是 `verbatim` 环境, 还是 `\verb` 命令都不能用作其他任何命令的参数。

有的时候, 我们可能需要一方面想照原样显示文本, 但另一方面又希望能执行文本中的命令。为达此目的, 我们可以上载 `alltt` 宏包 (第 204 页), 这个宏包提供了 `alltt` 环境, 环境文本中的命令和 `{}` 仍然保持正常情形中的功能。详见下例:

```
\begin{alltt}
    $x^2$ results \begin{math} x\sp2 \end{math}
\end{alltt}
```

结果为:

```
$x^2$ results   $x^2$ 
```

4.9 正文中的注释

在正文中若适当加上注释、解释、提示等等的文本, 对于提示作者或者告诉其他由于某一目的进行某种构造的用户而言, 是非常有用的。这些注释文本是要被格式化程序忽略掉的。

在 T_EX 中, 单个字符命令 `%` 就引进了注释。当这个字符出现在正文中, 它自己本身以及该行后面的其他文本都要被忽略。如果一个注释有几行长, 那么每行都必须前缀 `%`。

注释符号 `%` 对暂时抑制某些命令也是很有用的。把一个 `%` 放在某条命令的前面, 那么就会使该行后面的部分都被忽略。这称为 *注释抑制行*。类似地, 在源文件中可能有不同的替换文本, 因此可以注释掉一些行, 这样作者可以随时决定采用哪一个版本 (只要去掉要使用行前面的 `%` 符号, 且在其他行前面加上 `%` 符号就可以了)。

最后提一下, `%` 符号对去掉每行结尾的那个隐含空格有相当重要的作用。在用户定义中当希望在两个参数间不要由于分行而引进不必要的空格时, 就可以使用这个符号。

在调试 T_EX 和 L^AT_EX 源文件时, 如果某块文本隐藏着严重错误, 那么我们就需要采用分块隔离的方法, 找到出错的位置。一种常用的方法就是在认为正确的文本末尾插入

```
\end{document}
```

命令, 提前结束源文件。但这样会使得后面大批可能正确的文本也看不到。

还有一种情形, 在编辑文件时, 有时发现一长段文本暂时没有什么作用, 而以后有可能用到。如果每行前面都加上 `%`, 那可不是一件小工作量的事情。而且说不定将来想用这块文本, 又需要再去掉注释符号呢! 因此我们非常希望能有一种块注释的功能, 即把一段文本注释掉。为此, 我们要利用 T_EX 命令 `\iffalse` 和 `\fi`。从 `\iffalse` 开始, 到 `\fi` 结束的所有文本 (含这两命令自身) 都会被忽略。例如,


```

Comment out the changes%
\iffalse
in your preamble          等价于      Comment out the changes%
\fi                        . These commands may be ....
. These commands may be ....

```

结果都是： Comment out the changes. These commands may be

4.10 脚注与边注

4.10.1 标准脚注

脚注是用如下命令生成的：

```
\footnote{ 脚注文本 }
```

该命令紧接在需要在脚注中进行解释的单词后面。脚注文本 用较小的字样以脚注的形式显示在当前页的底部。脚注的第一行要缩进，并给出与正文插入点处相同的脚注标记。在一页上的第一个脚注是用一条短水平线与正文分开的。

标准脚注标记是一个小的偏上的数字¹，它是顺序编号的。这里为了得到脚注，采用如下输入：

```
... 小的偏上的数字 \footnote{ 在一篇用打字机打印的草稿中，通常是用...
有相同标记的问题。 }，它是顺...
```

在 `article` 类中，整个文档统一对脚注进行编号，而在 `report` 和 `book` 类中，每当开始新的一章时脚注的编号都会重新从 1 开始。

`\footnote` 脚注只可以位于通常的段落模式中，不能位于数学模式或 LR 模式 (2.1.3 节) 中。实际上，这就意味着它不能位于一个 LR 盒子 (4.7.1 节) 或者子段盒子 (4.7.3 节) 中。然而，它却可以用在 `minipage` 环境中，此时脚注文本显示在小页的下面，而不是实际页面的底部²。

`\footnote` 命令必须紧接在接受这个注释的单词后面，中间不必有任何空格或间隔。一句话的脚注可以在句号后面给出。

4.10.2 非标准脚注

如果用户希望在 `article` 类中每当开始新的一节时，脚注编号重置为 1，那可以用如下命令来做到：

```
\setcounter{footnote}{0}
```

只要把这条命令放在 `\section` 命令的前面或后面就可以了。

¹ 在一篇用打字机打印的草稿中，通常是用 *, ** 等等来标记脚注，这里也可以使用这种方法；然而，由于当输入文本时并不知道将在哪儿分页，因此这里就存在着如何避免在一页上有相同标记的问题。

² 对于嵌套的小页，脚注会在遇到第一个 `\end{minipage}` 命令时显示出来，这样就有可能使脚注出现在错误的地方。

内部脚注计数器的名称为 `footnote`。每次调用 `\footnote` 都会使这个计数器值增 1，并以阿拉伯数字的形式显示出当前值做为脚注的标记。可以用如下命令来实现不同样式的标记：

```
\renewcommand{\thefootnote}{\数字样式{footnote}}
```

这里的 `数字样式` 就是一个在 4.3.5 节中所描述的计数器显示命令：`\arabic`、`\roman`、`\Roman`、`\alph` 或 `\Alph`。对于 `footnote` 计数器，还可以使用另外一条计数器显示命令，它就是 `\fnsymbol`，这条命令把从 1 到 9 的计数器值显示为如下 9 个符号：

```
* † ‡ § ¶ || ** †† ‡‡
```

这里需要用户在调用第十个 `\footnote` 命令之前把脚注计数器重置为零。

在 `\footnote` 命令中还可以有一个可省参数：

```
\footnote[数]{脚注文本}
```

这里的 `数` 是一个正数，用它来取代显示标记的脚注计数器的值。在这种情形中，脚注计数器的值并没有增加。例如 **，

```
\renewcommand{\thefootnote}{\fnsymbol{footnote}}
```

例如 `\footnote[7]{第七个标记符号。}`，

```
\renewcommand{\thefootnote}{\arabic{footnote}}
```

这里的最后一行是为了把脚注标记样式恢复成标准形式。否则，后面所有的脚注都以符号为标记，而不是用数字。

4.10.3 禁止模式中的脚注

可以用如下模式在文本中插入一个脚注标记：

```
\footnotemark[数]
```

即使当前文本中不允许使用脚注，例如在 LR 盒子、表格和数学模式中，也可以使用这条命令。这里的标记就是相应于可省略的参数 `数` 或者当这个参数省略时脚注计数器增加后的值。这条命令并没有生成脚注，我们必须在禁止模式外面用如下命令补上实际的脚注：

```
\footnotetext[数]{脚注文本}
```

如果在脚注标记中已用了可省参数，那么在文本命令中也必须用相同的 `数` 做选项。类似地，如果在标记中没有使用可省参数，在文本命令中也不能出现可省参数。这里的脚注生成时需要利用 `数` 的值或者脚注计数器的值。

当 `\footnotemark` 命令没有可省参数时，每调用一次就会使脚注计数器的值增加 1，而相应的 `\footnotetext` 命令并不改变计数器的值。

如果在接下来的 `\footnotetext` 命令前面有许多 `\footnotemark` 命令，它们都没有带可省参数，那么就需要用如下命令调整计数器的值：

```
\addtocounter{footnote}{差}
```

这里的 `差` 是一个负数，它表示计数器必须被减少多少。因此在每次调用 `\footnotetext` 命令之前，还必须给计数器的值增 1。这就可以用 `差=1` 的 `\addtocounter` 命令或者

** 第七个标记符号。

```
\stepcounter{footnote}
```

命令来给计数器增 1。

For example: mosquitoes³ and elephants⁴

```
For example: \fbox{mosquitoes\footnotemark\ and
elephants\footnotemark}
```

生成脚注标记³和⁴。那么现在计数器的值为 4。为了使有框盒子外第一个 `\footnotetext` 相应于正确的计数器值，现在要把它减 1。这样两个脚注文本就可以如下生成：

```
\addtocounter{footnote}{-1} \footnotetext{Small insects}
\stepcounter{footnote}\footnotetext{Large mammals}
```

把它们就放在 `\fbox{}` 命令的后面。现在脚注计数器的值与它离开 `\fbox` 时的相同。

4.10.4 小页环境中的脚注

在 4.10.1 节中已提到，在小页环境中可以用脚注命令。然而脚注是显示在小页的下面，而不是当前页的底部。

在小页环境^a中的脚注命令有不同的标记样式。脚注会在接下来第一个的 `\end{minipage}` 命令时显示出来。^b小页环境中的脚注与正文所用的脚注使用不同的计数器，它的计数器名称为 `mpfootnote`，其记数方式与 `footnote` 无关。

得到左边结果的输入为：

```
\begin{minipage}{8cm}
```

在小页环境 `\footnote{` 标记是偏上的小写字母。`}` 中的脚注命令有不同的...

```
\end{minipage}
```

^a标记是偏上的小写字母。

^b当小页环境嵌套时，这有可能导致麻烦。

在表格即 `tabular` 环境中的脚注通常只能用上面所描述的命令得到：`\footnotemark` 放在表格内部，`\footnotetext` 放在环境外面。然而如果 `tabular` 环境是在一个小页环境内部，通常的 `\footnote` 命令也可以在表格内部使用。脚注显示在小页结束时的表格下方。

4.10.5 脚注样式参数

有两个脚注样式参数可以在需要时进行修改，这种修改既可以在导言中进行，也可以在一个环境内进行。

```
\footnotesep
```

两个脚注间的竖直距离。可以用 `\setlength` 命令修改这个长度。

```
\footnoterule
```

，这条命令在正文与脚注之间画一条水平线。它并不会增加任何竖直间距。可以修改它的定义，例如，

```
\renewcommand{\footnoterule}
```

³Small insects

⁴Large mammals

`{\rule{宽度}{高度}\vspace{-高度}}`

当高度的值为零时就会生成一个零高度的不可见直线。

4.10.6 边注

在页边上的注释可以用下面的命令生成：

`\marginpar{注释文本}`

它把注释文本放在右面的页边上，开始点与命令所在行相平。这里所出现的边注就是用 这是一个
如下输入得到的：

... 这里所出现的边注 `\marginpar{这是一个\\一个\\边注}` 就是 ... 边注

这里的注释文本通常放在一个宽度为 1.9cm(0.75in) 的子段盒子中。这样窄的盒子通常会使得断行非常困难，这也就是上面为什么要用 `\\` 命令人为断行的缘故。因此在边注中，在这样窄的盒子里只放一个符号就是非常合适的，如这里所显示的箭头。 ←

另一种需要使用边注的地方就是在要引起注意的文本旁边画一条竖线。这有时用来表示相对于以前的版本，这部分文本进行了改动。本段中的标记样例就是在第一行中用如下输入得到的：

`\begin{marginpar}{\rule[-17.5mm]{1mm}{20mm}}`

边注的宽度可以用下一节所描述的样式参数进行改动。用户必须保证总宽度不能变得超过打印机所接受的限度。

在默认状态下，边注出现在一页的右边，或者当选定了 `twoside` 选项时的外页边。这里的‘外’指的是奇数页的右页边，偶数页的左页边。当用了 `twocolumn` 选项时，它们就被放在外面的边界上：左列的左边，右列的右边。

但这样做有时会使类似上面给出的箭头这样的页边记号出现问题。如果该页为偶数页，它就必须指向相反的方向。事实上，它的方向是与它位于页的哪一边有关，也就是依赖于页码和列。由于在书写（或者后来对它进行修改）时并不知道它指向哪个方向，因此我们要利用 `\marginpar` 命令的扩充语法来实现这一点：

`\marginpar[左文本]{右文本}`

这种形式的命令包含页边文本的两个版本，左文本会出现在左边，右文本会出现在右边，具体视需要哪个面而定。因此为了全面起见，排版上面那个箭头的边注命令应该是

`\marginpar[\hfill\Longrightarrow]{${\Longleftarrow$}}`

（箭头命令是数学符号，因此需要放在 `$. $` 内，将在 5.3.4 节对它们进行详细介绍。）

在上面的 `\marginpar` 例子中若没有用 `\hfill` 命令，那么位于左边界上的箭头就会显示在其所在段的左边，从而离正文很远。之所以会这样，是因为 `\marginpar` 命令把它的内容左对齐处理，靠左边对齐正文的方式只适用于边注位于右边界的情形；如果它位于左边界，就会离正文太远，`\hfill` 命令就是为了使得边注内容向右边对齐，从而使得它处在相对于正文比较恰当的位置上。

对于这一节第一个边注，也需要用同样的方法来处理。实际生成它的命令是

`\marginpar[\flushright 这是一个\\一个\\边注]{这是一个\\一个\\边注}`

这里的 `\flushright`(4.2.2 节) 就等价于在每一行上放一个 `\hfill`。

边注的标准位置可以用 `\reversemarginpar` 命令来进行变换。一旦使用了这个命令, 边注就会位于左边, 或者在 `twoside` 选项时出现在内边界。这个命令的作用直到出现相反命令 `\normalmarginpar` 为止。对于 `\twocolumn` 命令, 这两个命令没有作用。

在边注中不能进行分页。如果一个边注太靠近页面底部, 因而没有足够空间放下它, 那么边注就会延伸到最后一行的下面。在这种情形中, 就需要在 `\marginpar` 命令中的文本开头处包含一条 `\vspace` 命令, 以使得边注向上移动一点儿, 或者就用两条 `\marginpar` 命令把它分成两部分, 分放在不同的页上。这种手工调整只有在整篇文档完成后才能进行, 因为以后对文档进行改动就会使情形发生变化。

4.10.7 边注的样式参数

可以改动下面的样式参数, 以重定义边注的显示方式:

`\marginparwidth`

定义边注盒子的宽度;

`\marginparsep`

设置边注盒子与正文边界之间的距离;

`\marginparpush`

两个边注盒子之间的最小竖直距离。

这些参数都是长度, 可以用 `\setlength` 命令像通常那样赋予一个新值。

第五章 数学公式的排版

排版数学公式是一件非常复杂的工作,在通常的打字机上根本无法进行。当初 Donald Knuth 之所以要设计 $\text{T}_{\text{E}}\text{X}$ 这一系统,就是为了解决排版数学公式这一难题。因此可以说数学公式的排版是 $\text{T}_{\text{E}}\text{X}$ 的灵魂。另一方面,我们知道 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 的核心是文档设计。但事实上, $\text{T}_{\text{E}}\text{X}$ 的所有数学排版功能 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 也都具备,而且进行了相当好的组合。我们在本章除了介绍 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 所提供的数学排版功能(实际上绝大部分内容都适用于 $\text{T}_{\text{E}}\text{X}$),另外还同时穿插介绍了 $\text{A}_{\text{M}}\text{S-}\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 中与数学排版有关的命令,这组命令增强了对复杂数学结构的控制,并添加了许多数学符号。

数学公式的排版,需要通过输入特殊的描述性文本来实现。这就意味着必须告诉 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 要把哪些文本解释成一个数学公式,而且也要告诉它数学公式什么时候结束,从而返回到正常的文本状态。数学文本的处理是通过切换进入数学模式(2.1.3 节)实现的。数学环境就是为了这个目标而引进的。

5.1 数学环境

数学公式可以有两种形式。一种是出现在正文的文本行中,如 $(a+b)(a-b) = a^2 - b^2$, 另一种就是与文本行分开,单独占一行,如

$$x = \frac{-2t^2}{1+t^2}, \quad y = \frac{2t}{1+t^2}.$$

我们分别称这两种形式的公式为 正文公式 和 单独公式。

正文公式是用如下环境生成的:

```
\begin{math}公式文本\end{math}
```

由于正文公式通常都很短,有的时候只有一个字符,因此也可以用一个更方便的版本,即 $\backslash(公式文本\backslash)$ 的形式来输入。如果还嫌它长,那可以用更短的形式 $\$公式文本\$$ 。所有这三种形式基本上是等价的,只是在内部具有某些小差别,如 $\backslash($ 是脆弱命令,而 $\$$ 就比较牢靠。

单独公式或方程是用下面的环境生成的:

```
\begin{displaymath}公式文本\end{displaymath}
```

```
\begin{equation}公式文本\end{equation}
```

这两种环境的差别在于 `equation` 环境会自动给公式加上一个顺序的公式编号。`displaymath` 环境还有一个更方便的等价形式 $\backslash[公式文本\backslash]$ 。同样如果还嫌它长,可以用另外的简短形式: $$$公式文本$$$ 。

在默认方式下,单独公式是水平居中的,而且如果有公式编号的话,编号会显示在右边。通过选择 `fleqn`(3.1.1 节)文档类选项,公式就会左对齐,而且可以具有一个能调整的

缩进。这个选项在整篇文档中有效，而缩进量可以用命令 `\setlength{\mathindent}{缩进量}` 来改变，这里的 缩进量 就是所要指定的缩进长度。除此之外，文档类选项 `leqno` 会使得整篇文档中公式编号显示在左边界。

无论是正文公式，还是单独公式，其中的 公式文本 都由数学结构组成，我们将在下面几节中介绍这些结构。

最后提一下，可以用如下环境创建多行公式：

`\begin{eqnarray}` 公式文本 `\end{eqnarray}`

`\begin{eqnarray*}` 公式文本 `\end{eqnarray*}`

这里的标准形式会给每行公式都加上一个顺序的公式编号，而 `*`- 形式则没有公式编号。另外在 $\mathcal{A}_{\mathcal{M}}\mathcal{S}\text{-}\text{\LaTeX}$ 中存在着许多环境或命令，以生成各种格式的多行公式，详情见 5.5 节。

5.2 数学公式的主要组成

5.2.1 常量与变量

在公式中的常量由数字组成，而简单变量则用一个字母表示。在数学排版中，一般用罗马字体显示常量，用斜体显示变量。 \LaTeX 在数学模式中会自动遵守这个规则。在源文本中为了方便以后研读而加上的空格都会被忽略。在常量、变量和类似于 $+$, $-$, $=$ 这样的运算符之间的距离是由 \LaTeX 自动设置的。例如 `$w=-u+7v$` 和 `$w = -u + 7v$` 都生成 $w = -u + 7v$ 。

有些数学符号在键盘上存在着对应的按键，它们是：

`+ - = < > / : ! ' | [] ()`

它们都可以直接用在数学公式中。大括号 `{ }` 用来表示公式的逻辑组合，因此不能作为可直接显示的字符。为了在公式中显示大括号，就必须与通常文本中一样使用 `\{` 和 `\}` 命令。例如：

`f(s) < f(t) < |f| = 1` `$f(s) < f(t) < |f| = 1$`
`k\{x_1, x_2\} \subset k[[x_1, x_2]]` `$k\{x_1, x_2\} \subset k[[x_1, x_2]]$`

5.2.2 上下标

在数学公式中经常可以见到上下标，以表示指数或指标。上下标要把字符相对于公式的主要文本所处的水平位置进行提升或下降，并以小号字体显示出来。甚至上下标本身还可以有上下标，这时只是再次进行提升或下降操作。

\LaTeX 和 \TeX 以一种简单的方式，提供了用适当字体尺寸创建任意指数和指标组合的方法：字符命令 `^` 把紧接下来的字符做为上标（提升），而字符命令 `_` 把紧接下来的字符做为下标（下降）。例如，

`z^2` `z^2` `b_n` `b_n` `B_i^n(x)` `B^n_i(x)`

当上下标一起出现时，它们的顺序是无关紧要的。上面最后那个例子也可以用 `B_i^n(x)` 来生成。

如果上标或下标的内容不只一个字符, 那么就必须用大括号 $\{ \}$ 把这组符号包围起来:

$$x^{2n} \quad x^{\{2n\}} \quad y_{i+j} \quad y_{\{i+j\}} \quad A_{i,j,k}^{-2n!} \quad A_{\{i,j,k\}}^{\{-2n!\}}$$

也可以在指数和指标中多次进行提升或下降操作:

$$x^{y^2} \quad x^{\{y^2\}} \quad x^{y_1} \quad x^{\{y_1\}} \\ B_{n+1}^{d_{i+j}-1} \quad B_{\{n+1\}}^{\{d_{\{i+j\}}-1\}}$$

注意: 提升和下降命令 \wedge 和 $_$ 只能用在数学模式中。如果在正文中需要一个做为上标或下标的符号, 例如 \dagger , 可以输入 $\${}\wedge\{\dagger\}$ 。另外 \sb 和 \sp 命令等价于这里的 $_$ 和 \wedge 。虽然很少用这两条命令, 但在 `alltt` 宏包及环境中就必不可少, 见 4.8 节。

5.2.3 分数

比较简单的分数, 尤其是正文公式中的分数最好用斜杠 $/$ 表示, 例如 $\$(n+m)/2\$$ 表示 $(n+m)/2$ 。对于较复杂的分数, 命令

`\frac{分子}{分母}`

可以用来把分子放在分母上面, 而且中间有一适当长度的水平线。

$$y = \frac{2t}{1+t^2} \quad \backslash[y = \frac{2t}{1+t^2}\backslash]$$

$$\frac{a^2 - b^2}{a + b} = a - b \quad \backslash[\frac{a^2 - b^2}{a+b} = a-b \backslash]$$

分数也可以嵌套至任何层次:

$$\frac{\frac{a}{x-y} + \frac{b}{x+y}}{1 + \frac{a-b}{a+b}} \quad \backslash[\frac{\frac{a}{x-y} + \frac{b}{x+y}}{1 + \frac{a-b}{a+b}} \backslash]$$

L^AT_EX 会把分数中的分数用较小的字样显示出来。在 5.6.2 节中介绍了当 L^AT_EX 的字样尺寸选择不理想时如何进行修改的方法。

一般情况下, 在正文公式中 `\frac` 命令生成的分数要小于在单独公式中生成的分数。通常情况下, 这种自动机制是相当有用的, 然而有时候我们希望能对分式的显示大小进行调整。为此 `amsmath` 宏包对 `\frac` 命令进行了一些修改。可以用 `\dfrac` 命令得到与单独公式中相同大小的分数, 而 `\tfrac` 生成正文公式中的分数。例如,

$$\backslash[\tfrac{3 + a^2}{4 + b} \quad \quad \quad \dfrac{3 + a^2}{4 + b}\backslash]$$

的结果为

$$\frac{3+a^2}{4+b} \quad \frac{3+a^2}{4+b}$$

另外, 在 `amsmath` 宏包中包含一个广义的分数命令, 其语法为

`\genfrac{左定界符}{右定界限}{分数线粗细}{数学样式}{分子}{分母}`

其中各参数的意义及取值为

- 左定界符 是公式左边的定界符, 空白表示不给出定界符。
- 右定界符 是公式右边的定界符, 空白表示不给出定界符。
- 分数线粗细 表示分数线的粗细, 空白表示普通粗细。
- 数学样式的取值及相应意义为

0 表示 `\displaystyle`,
 1 表示 `\textstyle`,
 2 表示 `\scriptstyle`,
 3 表示 `\scriptscriptstyle`.

如果不给出取值, 就根据当前上下文确定样式。

- 分子 与 分母 表示要显示的广义分数的上下两部分。

从这条命令的语法可知, 常用的 `\frac{分子}{分母}` 就等价于 `\genfrac{}{}{}{}{分子}{分母}`; 而上面提到的 `\dfrac` 与 `\tfrac` 命令就等价于 `\genfrac{}{}{}{0}{分子}{分母}` 与 `\genfrac{}{}{}{1}{分子}{分母}`。

下面给出几个例子。如果输入

```
$$
\genfrac{}{}{1pt}{}{a+b}{c} \quad \quad \quad \backslash quad
\genfrac{}{}{1.5pt}{}{a+b}{c} \quad \quad \quad \backslash quad
\genfrac{}{}{2pt}{}{a+b}{c} \quad \quad \quad \backslash quad
\genfrac{[]{}{}{0pt}{}{a+b}{c}
```

\$\$

就得到如下结果:

$$\frac{a+b}{c} \quad \frac{a+b}{c} \quad \frac{a+b}{c} \quad \left[\frac{a+b}{c} \right]$$

5.2.4 方根

方根是用如下命令显示的:

`\sqrt[开方数]{参数}`

例如: `$\sqrt[3]{27} = 3$` 生成 $\sqrt[3]{27} = 3$ 。如果不写可省参数 开方数, 就会生成平方根: `$\sqrt{144}=12$` 得到 $\sqrt{144} = 12$ 。

方根符号的尺寸和长度是自动与 参数 大小匹配的:

`$\sqrt{x^2 + y^2 - 2xy} = |x-y|$` 的结果为 $\sqrt{x^2 + y^2 - 2xy} = |x - y|$, 而 $\sqrt[n]{\frac{x^n - y^n}{1 + u^{2n}}}$ 的生成文本为 `\[\sqrt[n]{\frac{x^n - y^n}{1 + u^{2n}}}\]`

方根也可以嵌套至任意层次:

$$\sqrt{x + \sqrt{x + \sqrt{x}}} \quad \quad \quad \backslash[\sqrt{x + \sqrt{x + \sqrt{x}}}]$$

5.2.5 求和与积分

在数学公式中常用的求和与积分符号是用 `\sum` 与 `\int` 命令生成的, 根据其所在的是正文公式还是单独公式, 它们可以有不同尺寸。

`\ldots` ... 偏下的点 `\cdots` ... 中间点
`\vdots` : 竖直点 `\ddots` \cdot 对角点

以生成具有正确间距的点。最好地说明前两条命令差别的例子是: a_0, a_1, \dots, a_n 和 $a_0 + a_1 + \dots + a_n$, 它们分别是用 `a_0, a_1, \ldots, a_n` 和 `$a_0 + a_1 + \cdots + a_n$` 生成的。

`\ldots` 命令也可以用在普通的文本模式中, 而其余三条命令则只能用在数学模式中。在文本模式中, `\dots` 命令可以代替 `\ldots`, 两者的作用完全一样。在 `amsmath` 宏包中, 数学模式中的 `\dots` 命令在绝大多数情形下, 会自动帮你选择到底用居中的, 还是偏下的省略号。

5.3 数学符号

在数学文本中要使用相当多的符号, 但是只有相当少的一部分可以直接通过键盘输入而得到。L^AT_EX 提供了通常要用的可以想像到的绝大部分的数学符号。这些符号的名称都要加上前缀 `\`, 其名称来自于各自的数学含义。我们在附录 A.2 中列出了所有的数学符号, 在下面几节中对它们进行了分类讲解。

5.3.1 希腊字母

要得到希腊字母, 只需要在字母名称前面加上命令字符 `\` 就可以了。大写字母与小写字母的唯一区别在于, 要得到大写希腊字母, 字母名称的第一个字母需要大写。有些字母并没有列在附录 A.2.2 的清单中, 这是因为它与某个拉丁字母相同。例如, 大写的 ρ 与拉丁字母 P 相同, 所以就没有再为它定义相应命令。

在数学公式中的大写希腊字母通常用的是(直立)罗马字样。如果需要斜体字样, 可以用数学字体样式命令 `\mathnormal` 来得到:

`$\mathnormal{\Gamma\Pi\Phi}$` 生成 $\Gamma\Pi\Phi$ 。

这条命令只能用在 L^AT_EX 2_ε 中, 它取代了用在 L^AT_EX 2.09 中的数学字样声明 `\mit`, 原来的用法是 `$\mit \Gamma\Pi\Phi$`。另外, 如果上载了 `amsmath` 宏包, 上述命令失效。这时输入 `\var...` 形式的命令, 就可以得到斜体大写希腊字母, 例如 `\varGamma` 的结果为 Γ 。

希腊字母只能用在数学模式中。如果要用在普通文本中, 就必须用 `$...$` 把它们括起来。

5.3.2 花体字母

在数学公式中也可以用如下 26 个花体字母:

$A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z$

利用数学字体样式命令 `\mathcal` 可以得到相应的花体字母:

`$\mathcal{A,B,C,\dots,Z}$`

(在 L^AT_EX 2.09 中与之等价的声明为 `\cal`)。

如果用 `\usepackage{mathscr}{eucal}` 方式调用了 `eucal` 宏包, 则

`\mathscr{A,B,C,...,Z}`

的结果为

$A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z$

如果调用了 `eufrak` 宏包或者 `amssymb` 宏包, 可以输入 `\mathfrak{A,B,...,Z}` 得到:

$\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \mathfrak{D}, \mathfrak{E}, \mathfrak{F}, \mathfrak{G}, \mathfrak{H}, \mathfrak{I}, \mathfrak{J}, \mathfrak{K}, \mathfrak{L}, \mathfrak{M}, \mathfrak{N}, \mathfrak{O}, \mathfrak{P}, \mathfrak{Q}, \mathfrak{R}, \mathfrak{S}, \mathfrak{T}, \mathfrak{U}, \mathfrak{V}, \mathfrak{W}, \mathfrak{X}, \mathfrak{Y}, \mathfrak{Z}$

5.3.3 运算符

在数学公式中, 有很多运算符, 它们分别是

- **二元运算符** 把两个量通过一个符号组合起来, 生成一个新的量, 这个符号称为二元运算符。例如 $A \cap B$ 中的 \cap 。二元运算符的详细清单见附录 A.2.4。
- **关系运算符** 当要比较两个数学量时, 就要用关系运算符把它们连接起来。如 $x \leq y$ 中的 \leq 。有的关系运算符具有多个名称, 如 `\leq` 和 `\leqslant` 的结果都是 \leq 。

关系运算符的求反或否定在数学中是用一个斜杠贯穿符号而得到的: 如 $=$ 和 \neq 表示等于和不等于。为了得到相应符号的否定, 我们要在其名称前而加上 `\not`, 因此 `\not\in` 得到 \notin 。这同样也适用于键盘字符, 如 `\not=`, `\not>` 和 `\not<` 的结果为 \neq , \nless 和 \ngtr 。

但注意 `\not\in` 和 `\notin` 的结果并一样, 前者为 \notin , 后者则为 \notin 。后者比前者更好看些。另外, `\neq` 和 `\not=` 的结果都是 \neq 。

关系运算符的详细清单见附录 A.2.3。

5.3.4 箭头与指针

在数学文稿中通常会有箭头符号, 它也称为指针。例如 $\sin x/x \rightarrow 1, x \rightarrow 0$ 中的两个 \rightarrow 。注意有些箭头符号也是具有多个名称, 如 `\gets` 和 `\leftarrow` 的结果都是 \leftarrow 。而 `\Longleftarrow` 命令也可以用 `\iff` 来代替, 但是后者 (\iff) 与前者 (\Longleftarrow) 相比, 在箭头两边要有一点空白。

箭头与指针的详细清单见附录 A.2.5。

5.3.5 其他各类符号

L^AT_EX 不但包括了在数学文本中可能出现的符号, 而且还提供了其他一些符号, 如 \aleph , \diamond 等等。在附录 A.2.1 和 A.2.6 中列出了这些符号。

5.3.6 具有两种尺寸的符号

根据所处为正文公式还是单独公式, 有些符号会以不同的大小显示出来, 例如 `\sum` 在正文公式中结果为 \sum , 而在单独公式中则显示为 Σ 。我们在 5.2.5 节中已介绍了符号 `\sum` 和 `\int`。并演示了如何给它们加上下限; 同样, 用移位命令 `^` 和 `_` 也可以给具

有两种尺寸的符号加上下限。有些符号的上下限位置也会视所处为正文公式还是单独公式而发生变化。正如在 5.2.5 节中指出的那样，如果上下限被放在符号的旁边，可以用命令 `\limits` 来强迫上下限放在符号的上方和下方。类似地，当上下限的标准位置是上方和下方时，可以用相反命令 `\nolimits` 使得上下限只是位于符号的旁边。

$$\int_0^\infty \oint_0^\infty \quad \backslash [\ointint^\infty_0 \ointint\limits^\infty_0 \quad \backslash]$$

$$\prod_{\nu=0}^n \prod_{\nu=0}^n \quad \backslash [\prod^\infty_{\nu=0} \prod\nolimits^\infty_{\nu=0} \quad \backslash]$$

具有两种尺寸运算符的详细清单见附录 A.2.9。

5.3.7 函数名称

在数学公式中普遍使用的标准是用斜体表示变量，而用罗马字体表示函数名。如果我们在数学模式中只是简单地写出函数名 *sin* 或 *log*， \LaTeX 就会把认为它们是变量 *s i n* 和 *l o g*，从而显示为 *sin* 和 *log*。为了告诉 \LaTeX ，我们需要的是一个函数名，那就需要在函数名前而加上命令字符 `\`，例如 `\sin` 的输出为 *sin*。关于 \LaTeX 可接受的函数名清单请见附录 A.2.8。

在这些函数中有几个也可以在显示时带上（下）限。这只要在函数名后面接指标命令就可以了：如 `\lim_{x \to \infty}` 在正文模式中是 $\lim_{x \rightarrow \infty}$ ，而在显示模式中是 $\lim_{x \rightarrow \infty}$ 。

下面这些函数名可以用指标命令 `_` 加上一个下限：

`\det` `\gcd` `\inf` `\lim` `\liminf` `\limsup` `\max` `\min`
`\Pr` `\sup`

最后要提一点，函数命令 `\bmod` 和 `\pmod{参数}` 都生成函数 *mod*，但却是两种形式：

`$ 1234567 \bmod 89 = 48$` 1234567 mod 89 = 48
`$ y \pmod{a+b} $` *y* (mod *a + b*)

注意，如果在函数名后面接上 `\limits` 命令，那么就会同运算符中一样，把上下标看成上下限。不过，如果 `\limits` 前接的运算符或函数名不是 \LaTeX 系统提供的，而是用户定义的，那么在编译时就会出现错误。为此，需要借助于 `\mathop` 命令。

`\mathop{\mathrm{Limits}}\limits_{x \to 0}` $\text{Limits}_{x \rightarrow 0}$

5.3.8 数学重音

在数学模式中可以用下面这些数学重音：

`\hat{a}` `\hat{a}` `\breve{a}` `\grave{a}` `\grave{a}` `\bar{a}` `\bar{a}`
`\check{a}` `\check{a}` `\acute{a}` `\acute{a}` `\tilde{a}` `\tilde{a}` `\vec{a}`
`\dot{a}` `\dot{a}` `\ddot{a}` `\ddot{a}`

当要给字母 *i* 和 *j* 加重音时，应该去掉它们的点。为此，需要用符号 `\imath` 和 `\jmath` 代替直接字母输入，如

$$\vec{\imath} + \tilde{\jmath}$$

对于 `\hat` 和 `\tilde`, 还存在一种宽的变体, 名称分别是 `\widehat` 和 `\widetilde`. 这两种符号可以放在一个公式上:

$$\widehat{1-x} = \widetilde{-y}$$

数学重音的详细清单见附录 A.2.7. 另外, 在 1998 年 6 月, 新增加了一个数学重音命令 `\mathring`, 其相应于文本模式时的 `\r` 重音。

5.4 其他要素

在前几节中描述的数学要素对于构造下面这样复杂的数学公式已是绰绰有余了:

$$\lim_{x \rightarrow 0} \frac{\sqrt{1+x}-1}{x} = \lim_{x \rightarrow 0} \frac{(\sqrt{1+x}-1)(\sqrt{1+x}+1)}{x(\sqrt{1+x}+1)} = \lim_{x \rightarrow 0} \frac{1}{\sqrt{1+x}+1} = \frac{1}{2} \quad (5.1)$$

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0 \implies U_M = \frac{1}{4\pi} \oint_{\Sigma} \frac{1}{r} \frac{\partial U}{\partial n} ds - \frac{1}{4\pi} \oint_{\Sigma} \frac{\partial}{\partial n} \left(\frac{1}{r} \right) U ds \quad (5.2)$$

$$I(z) = \sin\left(\frac{\pi}{2}z^2\right) \sum_{n=0}^{\infty} \frac{(-1)^n \pi^{2n}}{1 \cdot 3 \cdots (4n+1)} z^{4n+1} - \cos\left(\frac{\pi}{2}z^2\right) \sum_{n=0}^{\infty} \frac{(-1)^n \pi^{2n+1}}{1 \cdot 3 \cdots (4n+3)} z^{4n+3} + \frac{z^2+1}{z-1} \quad (5.3)$$

按照从左到右的顺序看公式, 那么要得到生成它们的输入文本应该是没有什么困难的。例如, 最后一个公式就是用如下输入生成的:

```
\begin{equation}
I(z) = \sin( \frac{\pi}{2} z^2 ) \sum_{n=0}^{\infty}
\frac{ (-1)^n \pi^{2n} }{ 1 \cdot 3 \cdot \dots (4n+1) } z^{4n+1}
- \cos( \frac{\pi}{2} z^2 ) \sum_{n=0}^{\infty}
\frac{ (-1)^n \pi^{2n+1} }{ 1 \cdot 3 \cdot \dots (4n+3) } z^{4n+3}
+ \frac{z^2+1}{z-1}
\end{equation}
```

仔细地看一下公式 (5.3), 你会发现在 `\cos()` 和 `\sin()` 中的括号 `()` 应该再大一点。而且这个公式长度恰好等于行宽, 如果它再长一点儿, 那就需要在某个恰当的地方把它断开, 后面那部分要相对于前面一行适当地定位。到现在为止, 我们所学的数学要素还没有提供这方面的功能。即使是类似于如何在公式中包含普通文本这样简单的事情, 我们到现在也没讲。这一节的其他部分就是为了解决这些问题的。

最后要提一下, 有时候 `TeX` 选择的字体尺寸并不总能令人满意, 如公式 5.2 的最后一个积分中, 如果显示的是 $\partial \frac{1}{r}$, 就要比 $\partial_r \frac{1}{r}$ 好看得多。我们在 5.6 节中会讲解如何做到这一点, 同时介绍了其他的格式辅助工具, 例如怎样调整公式两部分之间的水平距离。

5.4.1 公式的编号

上面的公式不是用 `displaymath` 环境或者其简写形式 `\[...\]` 生成的, 而用的是 `equation` 环境, 这样它就会自动给公式加上编号。在文档类 `book` 和 `report` 中, 公式是在章内顺序编号的, 如上所示, 编号前面有章号, 并且把它们一起放在小括号 `()` 内。而对于文档类 `article`, 公式是相对于整篇文档编号的。可以重新定义编号的样式, 例如下述命令:

```
\renewcommand{\theequation}%
{\arabic{chapter}.\arabic{section}.\arabic{equation}}
```

会使公式编号成为 1.5.3 形式, 表示第一章第五节的第三个有编号的公式。这时我们就需要把公式编号局限在一节内, 即每开始新的一节, 自动重设为零。要想做到这一点, 可以在每节标题后插入命令 `\setcounter{equation}{0}`。实际上, 还有更省事的方式, 那就是在导言中插入如下指令:

```
\makeatletter
\@addtoreset{equation}{section}
\makeatother
```

其中第一条和第三条指令切换 `@` 字符的角色。因为在普通源文件中, `LaTeX` 认为 `@` 不是字母, 因此含 `@` 的内部命令就无法使用。如果用了 `amsmath` 宏包,

```
\numberwithin{equation}{section}
```

则可以得到同样的效果。

在默认状态下, 公式编号右对齐, 而且相对于公式竖直居中。如果在公式行上没有足够的空间显示编号, 那就把编号放在公式下一行的右边。如果选择了文档类选项 `leqno`, 那么整篇文档的公式编号就是左对齐的。

公式的自动编号意味着作者在写作时并不一定知道编号到底是多少。在 8.3.1 节中介绍的 `LaTeX` 的交叉索引系统, 讲解了如何引用章节编号 (8.3.3 节), 这也同样适用于公式编号。通过在 `equation` 环境中包含一条 `\label{引用名}` 命令, 我们可以在正文中用 `\ref{引用名}` 来显示还不知道的公式编号, 这里的引用名是一个关键词, 它是字母、数字和符号的任意组合。如果使用了 `amsmath` 宏包, 也可以用 `\eqref` 命令, 这条命令会自动把编号放在小括号内。

注: 在使用了 `amsmath` 宏包时, `equation` 环境有 `*` 形式, 这时它不给公式加编号。另外在 `equation` 或 `equation*` 环境中不能有空行。

5.4.2 括号符号的尺寸自动调整

在数学中经常包含括号, 而且一般是成对出现的, 用来包围公式的某部分。当显示的时候, 这些括号应该与被包围公式有相同的尺寸。为此 `LaTeX` 提供了一对命令

```
\left 左括号 部分公式 \right 右括号
```

用来实现这一点。把命令 `\left` 就放在左括号符号的前面, 而 `\right` 就放在右括号符号的前面。

$$\left[\int + \int \right]_{x=0}^{x=1}$$

这里的一对括号 `[]` 会根据被包围公式的大小自动调整其尺寸, 从而导致指数和指标也相应地长高和降低。

`\left` 和 `\right` 命令必须成对出现。每一个 `\left` 命令必须在后面某处有一个相应的 `\right` 命令。这种匹配也可以嵌套。第一个 `\left` 是与最后一个 `\right` 配对的; 接下来的 `\left` 与倒数第二个 `\right` 配对, 依次类推。在一个嵌套中必须有相同数目的 `\right` 和 `\left`。

相对应的左括号和右括号符号可以任意组合, 并不一定要是逻辑上的一对:

$$\vec{x} + \vec{y} + \vec{z} = \begin{pmatrix} a \\ b \end{pmatrix}$$

这种括号的配对当然是很特别的, 但是 \LaTeX 仍然接受这种组合:

有的时候公式中只有左括号或右括号, 而没有相配对的部分。然而即使是这种情形, `\left ... \right` 命令也必须成对出现, 只是用 `.` 来表示那个看不见的括号符号:

$$y = \begin{cases} -1 & : x < 0 \\ 0 & : x = 0 \\ +1 & : x > 0 \end{cases}$$

上面例子中的 `array` 环境在 6.6.1 节中进行介绍, 它生成数学模式中的表格。

`\left ... \right` 命令可以应用于下面这 22 种不同的符号上。它们是:

<code>(</code>	<code>(</code>	<code>)</code>	<code>)</code>	<code>[</code>	<code>\lfloor</code>	<code>]</code>	<code>\rfloor</code>
<code>[</code>	<code>[</code>	<code>]</code>	<code>]</code>	<code>[</code>	<code>\lceil</code>	<code>]</code>	<code>\rceil</code>
<code>{</code>	<code>\{</code>	<code>}</code>	<code>\}</code>	<code><</code>	<code>\angle</code>	<code>></code>	<code>\rangle</code>
<code> </code>	<code> </code>	<code>\ </code>	<code>\ </code>	<code>\uparrow</code>	<code>uparrow</code>	<code>\Uparrow</code>	
<code>/</code>	<code>/</code>	<code>\</code>	<code>\backslash</code>	<code>\downarrow</code>	<code>downarrow</code>	<code>\Downarrow</code>	
				<code>\updownarrow</code>	<code>updownarrow</code>	<code>\Updownarrow</code>	

例如, `\left| ... \right|` 就会生成两条根据所包围公式文本的大小自动调整高度的竖线。

5.4.3 公式中的普通文本

有时候需要在公式中包含一些普通文本, 例如 `and`, `or`, `if` 等词语。这时虽然处于数学模式中, 但我们需要切换回 LR 模式 (2.1.3 节和 4.7.1 节)。通过在公式中执行命令 `\mbox{普通文本}`, 并结合水平间距命令如 `\quad` 和 `\hspace` 等等, 就可以做到这一点。例如:

$$X_n = X_k \quad \text{if and only if} \quad Y_n = Y_k \quad \text{and} \quad Z_n = Z_k$$

输入文本为:

```
\[ X_n = X_k \quad \mbox{if and only if} \quad Y_n = Y_k \quad \text{and} \quad Z_n = Z_k \quad \]
```


`\end{array} \]`

根据表格构造要素的规定 (6.6.1 节), 我们知道 `@{文本}` 会在相邻两列间插入 文本的内容。在上面的例子中, 这就是 `\:+\:` 和 `\;=\;`。 `\:` 和 `\;` 命令现在还没有讲, 它们用来生成数学模式中小的水平间距 (5.6.1 节)。 `*{3}{c@{\:+\:}}` 是列定义 `c@{\:+\:}` 的三次重复。 `c` 规定了列文本是居中排列的。 `\multicolumn{5}{c}` 把五列进行了合并, 用一个居中条目代替。 `\dotfill` 用点充满该列。上面的方程组也可以用下面的简单格式生成:

`\begin{array}{c@{\:+\:}c@{\:+\:\cdots+};c@{\;=\;};c}`

也可以把 `array` 环境嵌套任意层次:

$$\left(\begin{array}{cc|c} x_{11} & x_{12} & x \\ x_{21} & x_{22} & y \end{array} \right)$$

```

\left( \begin{array}{c}
\left| \begin{array}{cc}
x_{11} & x_{12} \\
x_{21} & x_{22}
\end{array} \right| x \\
y
\end{array} \right)

```

绝大多数域中的列条目都是居中排列 (`c`) 的。在上面的列中第一个条目仍是一个域, 它具有两个居中的列。其左右用可自动调整高度的竖线包围。

`array` 环境在结构上与竖直盒子是一样的。这就是说在包围它的环境中, 它只是被做为单个字符处理, 因此它也可以同其他符号和结构一起出现:

$$\sum_{p_1 < p_2 < \dots < p_{n-k}}^{(1,2,\dots,n)} \Delta_{p_1 p_2 \dots p_{n-k}} \sum_{q_1 < q_2 < \dots < q_k} \begin{vmatrix} a_{q_1 q_1} & a_{q_1 q_2} & \dots & a_{q_1 q_k} \\ a_{q_2 q_1} & a_{q_2 q_2} & \dots & a_{q_2 q_k} \\ \dots & \dots & \dots & \dots \\ a_{q_k q_1} & a_{q_k q_2} & \dots & a_{q_k q_k} \end{vmatrix}$$

```

\left[ \sum_{p_1 < p_2 < \cdots < p_{n-k}}^{(1,2,\ldots,n)}
\Delta_{\begin{array}{l}
p_1 p_2 \cdots p_{n-k} \\
\end{array}}
\sum_{q_1 < q_2 < \cdots < q_k} \left| \begin{array}{llcl}
a_{q_1 q_1} & a_{q_1 q_2} & \cdots & a_{q_1 q_k} \\
a_{q_2 q_1} & a_{q_2 q_2} & \cdots & a_{q_2 q_k} \\
\multicolumn{4}{c}{\dotfill} \\
a_{q_k q_1} & a_{q_k q_2} & \cdots & a_{q_k q_k}
\end{array} \right|
\right]

```

在这个例子中, Δ 的下标用到了 `array` 环境。然而, 这个下标相对于其他部分公式显得太大了。在 5.4.7 节中提供了更好地解决这种域指标的方法。

同其他的表格环境和小页环境一样, `array` 环境也可以包含可省的竖直定位参数 `b` 或 `t`。在 4.7.3 和 6.6.1 节中已描述了其语法和结果。只有当域不需要竖直居中, 而要相对于其顶行或底行竖直定位时才会用到这个参数值。

$$\begin{array}{c}
 a_1 \\
 \vdots \\
 a_n \\
 u+v \quad -120
 \end{array}
 \begin{array}{c}
 -u-v \quad 10 \\
 \\
 12 \\
 -120
 \end{array}$$

```

\[\begin{array}{c}
a_1 \\
\vdots \\
a_n \\
u+v \quad -120
\end{array}
\begin{array}{c}
-u-v \quad 10 \\
\\
12 \\
-120
\end{array}
\]

```

我们建议读者借助于右边的生成文本, 尝试推断如何构造各种不同的域。另外在 $\text{T}_\text{E}_\text{X}$ 中提供了一条 `\bordermatrix{}` 命令, 从下面的例子, 我们就可以知道它的作用。

$$\begin{array}{c}
 C \quad I \quad C' \\
 C \begin{pmatrix} 1 & 0 & 0 \\ b & 1-b & 0 \\ 0 & a & 1-a \end{pmatrix}
 \end{array}$$

```

\bordermatrix{C & I & C' \\
C & 1&0&0 \\
I & b&1-b&0 \\
C' & 0&a&1-a}

```

5.4.5 公式上下的直线

命令

`\overline{部分公式}` 和 `\underline{部分公式}`

会在部分公式的上而或下面画一条直线。而且它们可以嵌套任意次:

$$\overline{\overline{a^2} + \underline{xy} + \overline{\overline{z}}}$$

```

\[\overline{\overline{a}^2 + \underline{xy} + \overline{\overline{z}}}
\]

```

`\underline` 命令也可用在文本模式中, 以生成下划线, 而 `\overline` 只能用在数学模式中。

与这两条命令完全类似地还有:

`\overbrace{部分公式}` 和 `\underbrace{部分公式}`

它们在部分公式的上方或下方放上一个水平的大括号:

$$\overbrace{a+b+c+d} \quad \overbrace{a + \underbrace{b+c} + d}$$

在单独公式中, 这些命令也可以有指数或指标。(升高的)指数被放在上括号的上方, 而(降低的)指标放在下括号的下方。

$$\underbrace{a + \overbrace{b + \cdots + y + z}^{123}}_{\alpha\beta\gamma}$$

```

\[\underbrace{a + \overbrace{b + \cdots + y + z}^{123}}_{\alpha\beta\gamma}
\]

```

另外,

```

\[\overleftarrow{a} \quad \quad \overrightarrow{aa} \quad \quad \quad

```

```
\overleftarrow{aaa} \quad \underleftarrow{aaaa} \quad
\underrightarrow{aaaaa} \quad \underleftarrow{aaaaaa}
\]
```

的结果为

$$\overleftarrow{a} \quad \overrightarrow{aa} \quad \overleftrightarrow{aaa} \quad \overleftarrow{aaaa} \quad \overrightarrow{aaaaa} \quad \overleftrightarrow{aaaaaa}$$

其中前两条是标准的 L^AT_EX 命令, 而后四条则是 amsmath 宏包所提供的。另外, amsmath 宏包中还有可伸展的箭头, 利用 \xleftarrow 或者 \xrightarrow 命令可以得到根据其上方或下方内容确定自己长度的箭头。例如,

```
\[
A \xrightarrow{\mbox{i-1}} B \xleftarrow[\alpha \rightarrow \beta]{\alpha} C \xleftarrow[\gamma]{\gamma} D \xleftarrow{} E
\]
```

的结果为:

$$A \xrightarrow{i-1} B \xleftarrow[\alpha \rightarrow \beta]{\alpha} C \xleftarrow[\gamma]{\gamma} D \xleftarrow{} E$$

5.4.6 堆积符号

命令

```
\stackrel{上部符号}{下部符号}
```

把上部符号居中放在下部符号的上方, 这里放在上方的符号要以较小的字样显示:

```
\vec{x} \stackrel{\mathrm{def}}{=} (x_1, \dots, x_n) \quad \$\vec{x} \stackrel{\mathrm{def}}{=} (x_1, \dots, x_n)$
A \stackrel{\alpha'}{\longrightarrow} B \stackrel{\beta'}{\longrightarrow} C \quad \$ A \stackrel{\alpha'}{\longrightarrow} B \dots$
```

结合数学字体尺寸命令 (5.6.2 节), 用这个命令可以构造新的符号。例如, < 和 = 的组合 $\stackrel{\textstyle<}{=}$ 结果为 \leq , 有些作者就希望 \leq 具有这种形状。如果这里不用命令 \textstyle, 那么结果是 \leq 。

5.4.7 其他的 T_EX 数学命令

T_EX 中的数学命令 \atop 和 \choose 也是非常有用的, 可以应用于所有的 L^AT_EX 文档。(事实上, 所有 T_EX 数学命令中除了 \eqalign, \eqalignno 和 \eqaligno 外, 都可以用于 L^AT_EX 文稿中。) 它们的语法是

```
{ 顶部公式 \atop 底部公式 }
{ 顶部公式 \choose 底部公式 }
```

这两条命令都生成一个类似于没有分数线的分数结构。对于 \choose 命令, 该结构被包围在小括号内 (在数学中称之为二项式系数)。

$$\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \quad \backslash [\{n+1 \choose k\} = \{n \choose k\} + \{n \choose k-1\} \backslash]$$

$$\prod_{j \geq 0} \left(\sum_{k \geq 0} a_{jk} z^k \right) = \sum_{n \geq 0} z^n \left(\sum_{\substack{k_0, k_1, \dots \geq 0 \\ k_0 + k_1 + \dots = n}} a_{0k_0} a_{1k_1} \dots \right)$$

`\[\prod_{j \geq 0} \left(\sum_{k \geq 0} a_{jk} z^k \right) =`
`\sum_{n \geq 0} z^n \left(\sum_{\genfrac{}{}{0pt}{}{k_0, k_1, \ldots \geq 0}{k_0 + k_1 + \ldots = n}} a_{0k_0} a_{1k_1} \ldots \right) \]`

利用 L^AT_EX 中的 array 环境也可以生成类似的结构:

`\begin{array}{c} 顶部行 \\ 底部行 \end{array}` (atop)
`\left(\begin{array}{c} 顶部行 \\ 底部行 \end{array} \right)` (choose)

array 结构与相应 T_EX 命令之间的差别在于前者总是以普通正文公式的样式和尺寸显示, 而后者会视所处公式的部分而改变尺寸。

比较如下:

$$\Delta_{\substack{p_1 p_2 \cdots p_{n-k} \\ p_1 p_2 \cdots p_{n-k}}} \quad \text{这里的下标是用 \atop 生成的}$$

$$\Delta_{\substack{p_1 p_2 \cdots p_{n-k} \\ p_1 p_2 \cdots p_{n-k}}} \quad \text{这里的下标是用 array 环境生成的}$$

上面的 T_EX 命令也可以用来生成正文公式中的小矩阵, 如 $\begin{pmatrix} a & b & c \\ l & m & n \end{pmatrix}$ 或者 $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. 这里的第一个矩阵是用

`\left(\begin{array}{ccc} a & b & c \\ l & m & n \end{array} \right)`

生成的, 而第二个是用

`\left(\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right)`

生成的。

注 1: 在 amsmath 宏包中, 有一个生成二项式系数的命令: `\binom`. 例如 `\binom{a}{b+c}` 的结果在正文中为 $\binom{a}{b+c}$, 在单独公式中为 $\binom{a}{b+c}$.

注 2: 如果使用了 amsmath 宏包, 就可以很容易地生成多行上下标或上下限。例如,

`\[\sum_{\substack{i < n \\ i \text{ even}}} x_i^2 \]`

的结果就是

$$\sum_{\substack{i < n \\ i \text{ even}}} x_i^2$$

当然也可以利用 amsmath 宏包中的 subarray 环境对多行上下标 (限) 进行更多的控制。例如:

`\[\sum_{\begin{array}{c} i < n \\ i \text{ even} \end{array}} x_i^2 \]`

的结果就是

$$\sum_{\substack{i < n \\ i \text{ even}}} x_i^2$$

5.4.8 多行公式

所谓多行公式,就是指一个公式长达几行,其中每行的关系符号(例如 = 或 ≤)是彼此竖直对齐的。为此,可以利用环境

`\begin{eqnarray}` 第一行 `\\` ... `\\` 第 n 行 `\end{eqnarray}`

`\begin{eqnarray*}` 第一行 `\\` ... `\\` 第 n 行 `\end{eqnarray*}`

排版显示模式中的长达几行的公式或方程。公式或方程中的行与行之间用 `\\` 分开。每行的形式为

左边公式 & 中间公式 & 右边公式 `\\`

在显示的时候, 左边公式在左边列中右对齐, 而右边公式在右边列中左对齐, 中间公式在中间列居中。列分隔符号 & 标志着公式中列的分隔。通常中间公式就是单个的数学符号, 即上面所说的关系运算符。因此每行公式的形式类似于 `\begin{array}{rcl}` ... `\end{array}` 环境中的行。

在 `array` 环境和 `eqnarray` 环境之间的差别是: 后者排版为单独公式。这也就是说, 对于那些具有两种尺寸的符号, 会取它们的大尺寸形式, 分数的分子和分母也是正常尺寸。另一方面, 对于 `array` 环境而言, 列条目被设为正文公式, 会选取那些符号的小尺寸形式, 分数的两部分也是以较小尺寸显示的。

在 `eqnarray` 环境的标准形式中, 会自动给公式加上有序的编号, 而在 `*`-形式中则没有编号。为了不给标准形式中某行公式加编号, 可以在该行公式的结束符 `\\` 前面插入命令 `\nonumber`。

可以在正文中用 `\ref{引用名}` 命令引用公式的编号, 这里的引用名就是要用 `\label{引用名}` 命令赋给某一行公式的关键词。详情请见 8.3.1 节。

例:

$$\begin{aligned} (x+y)(x-y) &= x^2 - xy + xy - y^2 \\ &= x^2 - y^2 \end{aligned} \tag{5.4}$$

$$(x+y)^2 = x^2 + 2xy + y^2 \tag{5.5}$$

```
\begin{eqnarray}
(x+y)(x-y) &= & x^2-xy+xy-y^2 \nonumber\\
&= & x^2 - y^2 \\
(x+y)^2 &= & x^2+2xy+y^2
\end{eqnarray}
```

$$\begin{aligned}
 x_n u_1 + \cdots + x_{n+t-1} u_t &= x_n u_1 + (a x_n + c) u_2 + \cdots \\
 &\quad + (a^{t-1} x_n + c(a^{t-2} + \cdots + 1)) u_t \\
 &= (u_1 + a u_2 + \cdots + a^{t-1} u_t) x_n + h(u_1, \dots, u_t)
 \end{aligned}$$

```

\begin{eqnarray*}
x_{nu_1} + \cdots + x_{\{n+t-1\}u_t} &= & x_{nu_1} + (ax_n + c)u_2 + \\
&& \cdots \\
&+ & \left(a^{\{t-1\}x_n} + c(a^{\{t-2\}} + \cdots + 1)\right)u_t \\
&= & (u_1 + au_2 + \cdots + a^{\{t-1\}u_t})x_n + h(u_1, \ldots, u_t)
\end{eqnarray*}

```

在这里我们对后一个例子做一些解释。在输入文件的第二行上有一个自动调整尺寸的 `\left(... \right)` 对，这样的对只能位于一行中，即不能把用行结束符 `\\` 把它们分开！也就是说如果多行公式中有自动调整尺寸的括号，配对的 `\left` 和 `\right` 命令只能位于同一行中。

如果配对括号必须位于不同行中，那么可以尝试用

```
\left( ... \right. \quad \quad \left. ... \right)
```

结构。在第一行，`\left(` 与看不见的括号 `\right.` 配对，而第二行则由 `\left.` 开始，它与闭括号 `\right)` 配对。然而，这种方法只有当两部分公式的高度大致相当时才会使两个自动调整尺寸的括号的大小差不多。在 5.6.3 节中会告诉你当自动调整尺寸失败时如何手工选择括号的尺寸。

第二行开头的 `+` 也需要加以解释。在数学中 `+` 和 `-` 有两种意义：在两个数学量之间，扮演一种结合的角色（二元运算符），但如果只是在一个数学符号前，那它就是一个符号标志（正号或负号）。 \LaTeX 通过插入不同的间距来强调这种不同，请注意 `+b` 和 `a+b` 的差别。

$$\begin{aligned}
 y &= a + b + c + d \\
 &\quad + e + f + g \\
 &\quad + h + i + j
 \end{aligned}$$

如果一个很长的公式被分成几行，其中有的行以 `+` 或 `-` 开头， \LaTeX 会认为它是一个符号标志，从而把它向下一个字符移近。

解决的方法是在该行开头处引进一个零宽度的看不见字符。这可以是一个空结构 `\{ }`。请比较上面公式中 `&\ +e+f+g` 和 `&\{\}+h+i+j` 的差异。

有的时候对于多行公式最好采用下列这种断行的方法：

$$w + x + y + z =$$

$$\begin{aligned}
 &a + b + c + d + e + f + \\
 &g + h + i + j + k + l
 \end{aligned}$$

即第二行及其后续各行并不是相对于第一行的等号对齐，而是相对于第一行向右缩进一点后左对齐。

```
\begin{eqnarray*}
\lefteqn{w+x+y+z = } \\
& \& a+b+c+d+e+f+ \\
& \& g+h+i+j+k+l
\end{eqnarray*}
```

在第一行的命令 `\lefteqn{w+x+y+z =}` 可以使
得参数值的内容被显示出来, 但是 \LaTeX 认为它的
宽度为零。因此左边列只有列间距, 由它生成其余
各行的左面缩进。

可以通过在 `\lefteqn{...}` 和行结束符之间插入 `\hspace{深度}` 命令来改变缩进深度。正的 深度 会增加缩进, 而负值则减少缩进。

对那些有很多组逻辑括号的长公式, 尤其是嵌套很多次的情形, 刚开始排版时总会有错误。原因就在于括号的顺序不对或者遗漏了配对。如果在公式处理过程中 \LaTeX 报告有错误, 初学者经常无法理解错误信息, 这时候就应该仔细地检查公式文本中括号的配对。有些文本编辑器可以帮助用户搜索配对括号, 从而可以简化这种操作。

如果这样还没有找到错误所在, 那么用户就可以尝试, 当出现错误时按回车键, 以继续处理下去。通常浏览这样得到的显示结果中就会找到错误所在。

5.4.9 有框公式和并列公式

单独公式或方程也可以放在适当宽度的竖直盒子中, 即 `\parbox` 命令的参数或者 `minipage` 环境中。在竖直盒子中, 公式是水平居中的, 或者根据所选择的文档类选项进行 `\mathindent` 缩进后左对齐。

竖直盒子可以同单个字符一样, 相对于另一个竖直盒子定位 (4.7.3 和 4.7.7 节)。通过这种方法, 我们可以使得单独公式或方程并列摆放。

$\alpha = f(z)$ (5.6)	$x = \alpha^2 - \beta^2$	两个公式中左面的公式放在宽度为 4cm 的 <code>\parbox</code> 中, 而右边那个放
$\beta = f(z^2)$ (5.7)	$y = 2\alpha\beta$	在宽度为 2.5cm 的 <code>\parbox</code> 中,
$\gamma = f(z^3)$ (5.8)		本段文本则是在宽度为 5.5cm 的 <code>minipage</code> 中。

```
\parbox{4cm}{\begin{eqnarray} \alpha \&= f(z) ... \end{eqnarray}}
\hfill \parbox{2.5cm}{\begin{eqnarray*}
x \&= \alpha^2 - \beta^2 \\
y \&= 2\alpha\beta
\end{eqnarray*}}
\hfill \begin{minipage}{5.5cm} 左面的公式 ... \end{minipage}
```

竖直盒子也可以用来解决公式编号位置不恰当的问题。在 `eqnarray` 环境中 \LaTeX 为每行公式生成一个编号, 如果不希望有编号的话, 可以用 `\nonumber` 命令把它去掉。而为了给一组公式加上竖直居中的编号, 如:

$$\begin{aligned}
P(x) &= a_0 + a_1x + a_2x^2 + \cdots + a_nx^n \\
P(-x) &= a_0 - a_1x + a_2x^2 - \cdots + (-1)^n a_nx^n
\end{aligned} \tag{5.9}$$

就可以用如下输入来得到:

```
\parbox{10cm}{\begin{eqnarray*} ... \end{eqnarray*}} \hfill
\parbox{1cm}{\begin{eqnarray}\end{eqnarray}}
```


这里真正的方程是用 `eqnarray*` 环境生成的, 放在宽度为 10cm 的竖直盒子中, 其后接一个空的 `eqnarray` 环境, 这是一个宽度为 1cm 的盒子, 由它生成公式编号。这两个盒子都沿着它们自己的中间位置竖直对齐。

如果要用方框强调某公式, 并不需要什么新的结构要素, 只要把它们放在 `\fbox`(4.7.7 节) 中就可以了。正文公式 $a+b$ 要加上方框, 只要输入 `\fbox{$a+b$}` 就可以了。

要给单独公式加上方框, 那就首先需要把它放在一个宽度足够的 `\parbox` 或 `minipage` 环境中, 然后把它们放在一个 `\fbox` 中。(另外的方法见 5.6.6 节。)

$$\int_0^{\infty} f(x) dx \approx \sum_{i=1}^n w_i e^{x_i} f(x_i)$$

上面这个公式就是用如下输入生成的:

```
\fbox{\parbox{5cm}{\[\int^{\infty}_0 f(x)\, dx \approx ... \]}}
```

5.4.10 化学方程式和数学公式中的黑体

数学中有时需要把单个字符或者部分公式设成黑体。要做到这一点, 只需要利用在 5.4.3 节中提到的数学字体命令 `\mathbf` 就可以了:

```
$\mathbf{S^{-1}}TS = \mathrm{dg}(\omega_1, \ldots, \omega_n) = \Lambda$
```

可以得到 $\mathbf{S^{-1}}TS = \mathrm{dg}(\omega_1, \dots, \omega_n) = \Lambda$ 。

在这个例子中, 所有公式都被设为 `\mathbf` 的参数值, 这样全部都是黑体。但实际上只有数字、小写和大写的拉丁字母以及大写的希腊字母被 `\mathbf` 设成黑体。而小写的希腊字母和其他数学符号仍然是普通数学字体。

如果只希望一部分公式被设成黑体, 那么就必须把这部分公式放在 `\mathbf` 的参数中:

```
$\mathbf{2\sqrt{x}/y} = z$       $2\sqrt{x}/y = z$ 
```

数学字体命令 `\boldmath` 会把所有字符设成黑体, 只有下列符号例外:

- 上升或下降的符号 (指数和指标)
- 字符 `+ : ; ! ? () []`
- 有两种尺寸的符号 (5.3.6 节)

但是要注意, `\boldmath` 命令不能位于数学模式中, 必须在切换进入数学模式之前或者在子段盒子及小页环境中使用。相反的命令 `\unboldmath` 把数学字体重设回正常的字样。

```
\boldmath \[\oint\limits_C V\,d\tau = \oint\limits_{\Sigma} \nabla \times V\,d\sigma \]
\unboldmath
```

即使在数学模式外面已经使用了 `\boldmath`, 也可以在内部用如下命令暂时性地关闭这种设置: `\mbox{\unboldmath$...$}`, 从而把这部的公式以正常数学字体显示出来。

```
\boldmath\(\mathbf{P} = \mbox{\unboldmath$m$}b\) \unboldmath
```

的结果为: $P = mb$. 类似地, 在数学模式中也可用 `\mbox{\boldmath$...$}` 结构暂时性打开 `\boldmath` 设置: $W_r = \int M d\varphi = r^2 m \omega^2 / 2$ 的输入文本为:

```
\( W_r = \int \mbox{\boldmath$M$,d\varphi} = ... \)
```

化学方程式通常采用的是罗马字体, 而不是数学公式中的斜体。通过把公式作为 `\mathrm` 字体命令的参数值可以实现这一效果:

```
$\mathrm{Fe_2^{2+}Cr_2O_4}$  $\text{Fe}_2^{2+}\text{Cr}_2\text{O}_4$ 
```

如果仔细观察, 你会发现 Cr 和 O 的下标位置没有 Fe 的下标位置低。7.3.4 节在 167 页的例 5 中描述了如何简单地生成化学方程式, 而且不会有这种缺陷。

在 $\text{\LaTeX}2.09$ 中, 没有字体命令 `\mathbf` 和 `\mathrm`, 而是如 5.4.3 节所示, 用 `\bf` 和 `\rm` 取代。

5.5 \LaTeX - \LaTeX 对多行公式对齐功能的增强

前面介绍的标准 \LaTeX 对多行公式的排版功能, 应付一般的情况还是可以的。但是有些公式, 尤其是对齐设置相当复杂的公式, 排版起来就非常困难了。我们在本节主要介绍 \LaTeX - \LaTeX 中为多行公式对齐功能所提供的一些新环境。为了简单起见, 我们只是列出环境名, 并给出一个典型示例, 不讨论环境的详细语法。在掌握了前一节内容的基础上, 对新环境的理解应该没有什么大问题。

5.5.1 公式集合

可以用 `gather` 环境把多个单独公式汇集在一起, 例如,

```
\begin{gather}
x_1 x_2 + x_1^2 x_2^2 + x_3 \\\
x_1 x_3 + x_1^2 x_3^2 + x_2 \\\
x_1 x_2 x_3 \tag{222}
\end{gather}
```

的结果为:

$$x_1 x_2 + x_1^2 x_2^2 + x_3 \quad (5.10)$$

$$x_1 x_3 + x_1^2 x_3^2 + x_2 \quad (5.11)$$

$$x_1 x_2 x_3 \quad (222)$$

这里用 `\\` 把各个公式分开, 最后一行末尾没有 `\\`。每行都有公式编号, 但是如果用了 `\tag` 命令, 那么就会用 `\tag` 的参数做为标志。如果不想有编号, 可以使用 `\nonumber` 或 `\notag` 命令。在 `gather` 环境中不允许有空行。

`gather*` 环境与 `gather` 类似, 只是所有公式都不加编号, 但可以用 `\tag` 来加标志。

5.5.2 分割长公式

`multline` 环境可以把一个很长的公式分成几行, 第一行左对齐, 中间的几行居中, 最后一行右对齐。

```

\begin{multline}
(x_1 x_2 x_3 x_4 x_5 x_6)^2 \\\
+ (x_1 x_2 x_3 x_4 x_5 + x_1 x_3 x_4 x_5 x_6 \\
+ x_1 x_2 x_4 x_5 x_6 + x_1 x_2 x_3 x_5 x_6)^2 \\\
+ (x_1 x_2 x_3 x_4 x_7 + x_1 x_3 x_4 x_5 x_7 \\
+ x_1 x_2 x_4 x_5 x_7 + x_1 x_2 x_3 x_5 x_7)^2 \\\
+ (x_1 x_2 x_3 x_4 + x_1 x_2 x_3 x_5 \\
+ x_1 x_2 x_4 x_5 + x_1 x_3 x_4 x_5)^2
\end{multline}

```

的结果为:

$$\begin{aligned}
 & (x_1 x_2 x_3 x_4 x_5 x_6)^2 \\
 & + (x_1 x_2 x_3 x_4 x_5 + x_1 x_3 x_4 x_5 x_6 + x_1 x_2 x_4 x_5 x_6 + x_1 x_2 x_3 x_5 x_6)^2 \\
 & + (x_1 x_2 x_3 x_4 x_7 + x_1 x_3 x_4 x_5 x_7 + x_1 x_2 x_4 x_5 x_7 + x_1 x_2 x_3 x_5 x_7)^2 \\
 & + (x_1 x_2 x_3 x_4 + x_1 x_2 x_3 x_5 + x_1 x_2 x_4 x_5 + x_1 x_3 x_4 x_5)^2 \quad (5.12)
 \end{aligned}$$

这里也同样用 `\\` 分开各行, 整个公式有一个编号, 可以用 `\tag` 命令进行修改, 或者用 `\nonumber`, `\notag` 命令去掉编号。注意环境名为 `multline`, 而不是 `multiline`。

在 `multline*` 环境中, 公式没有编号, 但仍然可以用 `\tag` 命令加标志。第一行的缩进, 以及最后一行到页边的距离, 由 `\multlinegap` 参数确定, 其缺省值为 10pt, 可以用 `\setlength` 命令改变它的值。

在 `multline` 或 `multline*` 环境中的公式, 为了改变其缺省的对齐方式, 可以在相应行中加上 `\shoveleft` 或者 `\shoveright` 命令, 使之左对齐或者右对齐。

5.5.3 列的对齐

大多数的多行公式都要被分成不同的列, 在前面介绍的 `eqnarray` 和 `eqnarray*` 环境是把公式分成三列。如果有太多的列, 那么我们就必须借助于 $\mathcal{A}_{\mathcal{M}}S$ 的环境了。

利用 $\mathcal{A}_{\mathcal{M}}S$ - \LaTeX 的 `align` 环境, 可以很容易地排版有复杂对齐关系的多行公式。这时列数只受文本宽度限制。下面这个例子中, 有两个对齐列:

$$\begin{array}{ll}
 f(x) = x + yz & g(x) = x + y + z \\
 h(x) = xy + xz + yz & k(x) = (x + y)(y + z)
 \end{array}$$

输入文本为

```

\begin{align*}
f(x) &= x + yz & g(x) &= x + y + z \\\
h(x) &= xy + xz + yz & k(x) &= (x + y)(y + z)
\end{align*}

```

其中每行有三个 & 符号, 其第一个表示第一列的对齐点, 第二个表示列的分隔, 而最后一个表示第二列的对齐点。因此一般地说, 如果有 n 列, 那么每行中应该有 $2n - 1$ 个 & 符号。

align 环境有两个变体, 一个是 flalign 环境, 它把最左边的列靠左页边摆放, 最右边的列靠右页边摆放。要得到下面所示的结果, 只要把上面输入文本中的 align 环境用 flalign 环境替换就可以了。

$$\begin{array}{ll} f(x) = x + yz & g(x) = x + y + z \\ h(x) = xy + xz + yz & k(x) = (x + y)(y + z) \end{array}$$

另一个变体是 alignat 环境, align 环境会自动帮你计算要在列与列之间插入多大的空档, 而 alignat 环境在列与列之间没有任何空档, 这样用户就可以更好地控制空档大小。注意在 alignat 环境中有一个参数, 表示列数。

```
\begin{alignat*}{2}
  f(x) &= x + yz & & g(x) &= x + y + z & \\
  h(x) &= xy + xz + yz & & k(x) &= (x + y)(y + z) & \\
\end{alignat*}
```

$$\begin{array}{ll} f(x) = x + yz & g(x) = x + y + z \\ h(x) = xy + xz + yz & k(x) = (x + y)(y + z) \end{array}$$

为了增加列间距, 我们需要用两列之间插入 \quad 或者 \qquad 等间距命令。

```
\begin{alignat*}{2}
  f(x) &= x + yz & & \qquad g(x) &= x + y + z & \\
  h(x) &= xy + xz + yz & & \qquad k(x) &= (x + y)(y + z) & \\
\end{alignat*}
```

$$\begin{array}{ll} f(x) = x + yz & g(x) = x + y + z \\ h(x) = xy + xz + yz & k(x) = (x + y)(y + z) \end{array}$$

在排版多行公式时, 我们有可能需要在两行公式之间插入一行文本, 而且不希望该文本行与其他的公式对齐, 这时可以利用 \intertext 命令。

```
\begin{align}
  h(x) &= \int \left( \frac{f(x)+g(x)}{1+f^2(x)} + \right. \\
  &\quad \left. \frac{1+f(x)g(x)}{\sqrt{1-\sin x}} \right) \, dx \\
  \intertext{The reader may find the following form easier to read:}
  &= \int \frac{1+f(x)}{1+g(x)} \, dx - 2 \arctan(x-2) \notag
\end{align}
```

$$h(x) = \int \left(\frac{f(x) + g(x)}{1 + f^2(x)} + \frac{1 + f(x)g(x)}{\sqrt{1 - \sin x}} \right) dx \quad (5.13)$$

The reader may find the following form easier to read:

$$= \int \frac{1 + f(x)}{1 + g(x)} dx - 2 \arctan(x - 2)$$

5.5.4 大公式中部分公式的对齐

为了获得大公式中部分公式的对齐，如

$$\begin{array}{ll} x = 3 + a + \alpha & x = 5 + a + \alpha \\ y = 4 + b & y = 12 \\ z = 5 + c & z = 13 \\ u = 6 + d & u = 11 + d \end{array} \quad \text{or}$$

我们需要利用 `aligned` 和 `gathered` 环境，它们是 `align` 和 `gather` 环境的变体，语法也完全一样，区别就在于前者可以构造单独公式中的一个“大”数学符号。因此为了得到上面的输出，输入文本应该为

```
\[
\begin{aligned}
x &= 3 + a + \alpha \\
y &= 4 + b \\
z &= 5 + c \\
u &= 6 + d
\end{aligned}
\text{\quad or \quad}
\begin{gathered}
x = 5 + a + \alpha \\
y = 12 \\
z = 13 \\
u = 11 + d
\end{gathered}
\]
```

另外，`aligned` 和 `gathered` 都有一个可省的定位参数，其紧接在 `\begin{aligned}` 等命令的后面，形式为 `[c]`、`[t]` 或 `[b]`，分别表示中间、顶部或底部对齐。`[c]` 为默认值。

这里提一下，缺省情况中， $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$ 不会在长公式内分页，这样就有可能造成页的不满或过满情形。为此可以在调用命令 `\allowdisplaybreaks`，允许在长的单独公式内分页，在一定程度中解决这种问题。注意 `\allowdisplaybreaks` 是 `amsmath` 中的命令。

5.5.5 交换图的构造

利用 `amscd` 宏包, 可以绘制代数几何与交换代数等数学分支中经常用到的交换图。例如, 为了得到

$$\begin{array}{ccc} A & \longrightarrow & B \\ \downarrow & & \downarrow \\ C & \xlongequal{\quad} & D \end{array}$$

可以输入

```
\[
\begin{CD}
A @>>> B \\
@VVV @VVV \\
C @= D
\end{CD}
\]
```

从上面这个例子可见, 交换图由两种类型的行组成, 一种是水平行, 即行中有水平箭头; 另一种是竖直行, 即行中有竖直箭头。例如,

$$A @>>> B$$

就是一个典型的水平行。这定义两列, 中间用 `@>>>` 生成的箭头连接起来。当然可以有多个列, 例如,

$$A @>>> B @>>> C @= D @<<< E @<<< F$$

其中连接部分可以是 `@>>>` 和 `@<<<` (生成长度可变的箭头), 也可以是 `@=` (生成长度可变的等号)。在箭头上部的标签应该输入在第一个和第二个 `>` (或 `<`) 符号之间, 而箭头下方的标签应输放在后两个 `>` (或 `<`) 之间。

下面给出的是一个典型的竖直行组成:

$$@VVV @VVV @AAA$$

`@VVV` 给出一个向下的箭头, 而 `@AAA` 给出的一个向上的箭头。箭头左边的标签应输入在前两个 `V` (或 `A`) 之间, 而箭头右边的箭头应输入在后两个 `V` (或 `A`) 之间。竖直箭头是从第一列开始安置的。

例如,

$$\begin{array}{ccccc} C & \xrightarrow{H_1} & C & \xrightarrow{H_2} & C \\ P_{c,3} \downarrow & & P_{t,3} \downarrow & & \downarrow P_{-c,3} \\ C & \xrightarrow{H_1} & C & \xrightarrow{H_2} & C \end{array}$$

的输入文本为

```
\[
\begin{CD}
C @>H_1>> C @>H_2>> C \\
@V P_{c,3} VV @V P_{t,3} VV @VV P_{-c,3} V \\
C @>H_1>> C @>H_2>> C
\end{CD}
```

```

\mathbb{C} @>H_{1}>> \mathbb{C} @>H_{2}>> \mathbb{C} \\
@VP_{c,3}VV @VP_{\bar{c},3}VV @VVP_{-c,3}V \\
\mathbb{C} @>H_{1}>> \mathbb{C} @>H_{2}>> \mathbb{C} \\
\end{CD}
\]

```

如果想后移几个制表位，可以使用 `&` 命令。例如，

```

\[ \begin{CD}
A @>>> B \\ @V \end{CD} \quad \& \quad @VVV \\ C @= D \\
\end{CD} \quad \quad \quad \& \quad @VVV \\ C @= D \\
\end{CD}
\]

```

的输出为

$$\begin{array}{ccc}
 A & \longrightarrow & B \\
 \downarrow & & \downarrow \\
 C & \xlongequal{\quad} & D
 \end{array}
 \quad \quad \quad
 \begin{array}{ccc}
 A & \longrightarrow & B \\
 \downarrow & & \downarrow \\
 C & \xlongequal{\quad} & D
 \end{array}$$

注意上例中 `&` 命令的个数与对齐的关系。在第 31 页上有一个更复杂的交换图表示例。

5.6 数学公式的精调

至此我们给出的数学要素，几乎可以排版普通文稿中所可能出现的全部公式，前提条件是作者要遵守印刷数学公式时某种一般性的广为接受的规则。那些到现在为止还在用打字机打印文稿的作者可能会觉得由 \TeX 选择的字符间距太窄了。事实上，关于数学排版， \TeX 要比绝大多数作者知道得都多。在采用下面这节中的格式化工具进行修改公式排版时，作者应该首先去查看一些出版文献中类似公式的排版。否则，就有可能你费了半天事，用手工精心调整的 \LaTeX 公式，最后还是被专业制版工人重新改回原来标准 \TeX 所排版出来的样子。

5.6.1 水平间距

虽然 \TeX 对数学排版的规则有深入的了解，但是我们不能期望它理解数学公式的含意。例如 ydx 通常意味着变量 y 与微分算子 dx 的组合，这种组合的标志是两者之间有一个小间距。然而， \TeX 会去掉 $y\,dx$ 中的空格，从而显示为 ydx ，即 y , d 和 x 三个量的乘积。对于这种情形，我们就需要对 \LaTeX 进行一些精调。

在数学模式中可以用下面的命令插入很小的水平间距：

$\backslash,$ 小间距 = 3/18 of a quad
 $\backslash:$ 中间距 = 4/18 of a quad
 $\backslash;$ 大间距 = 5/18 of a quad
 $\backslash!$ 负间距 = -3/18 of a quad

在下面的例子中, 第三列就是没有插入水平间距命令的结果:

$\backslash\sqrt{2}\backslash,x$	$\sqrt{2}x$	$\sqrt{2}x$
$\backslash\sqrt{\backslash,\backslash\log x}$	$\sqrt{\log x}$	$\sqrt{\log x}$
$\backslash O\left(1/\sqrt{n}\right)$	$O(1/\sqrt{n})$	$O(1/\sqrt{n})$
$\backslash[0,1]$	$[0,1]$	$[0,1]$
$\backslash\log n\backslash,(\backslash\log\log n)^2$	$\log n(\log \log n)^2$	$\log n(\log \log n)^2$
$x^2\backslash!/2$	$x^2/2$	$x^2/2$
$n\backslash!\backslash\log n$	$n/\log n$	$n/\log n$
$\backslash\Gamma_2+\backslash\Delta^2$	$\Gamma_2 + \Delta^2$	$\Gamma_2 + \Delta^2$
$R_i^j\backslash_{kl}$	$R_i^j{}_{kl}$	$R_i^j{}_{kl}$
$\backslash\int_0^x\backslash!\backslash\int_0^y dF(u,v)$	$\int_0^x\int_0^y dF(u,v)$	$\int_0^x\int_0^y dF(u,v)$
$\backslash[\backslash\int\backslash!\backslash!\backslash!\backslash\int_D dx\backslash,dy \backslash]$	$\iint_D dx dy$	$\int\int_D dx dy$

注意: 在倒数第三个例子中的 $R_i^j\backslash_{kl}$, 在 R_i 的指标之后构造一个零宽度的不可见字符, 用这个空字符接受后面的指数。这样结果就是 R_i^j , 而不是 R_i^j 的 R_i^j 。

对于该在何时加上数学间距命令, 并没有一个严格的普遍适用的规则。但是当遇到前面提到的微分算子、正文公式中的小根号后接一个变量、除号 /、以及多重积分号等情形时, 就有必要考虑进行间距调整。上面的例子给出了许多需要精调的情形。

5.6.2 在公式中选择字体尺寸

我们可以改变 TeX 为各部分公式所选择的字体尺寸。首先我们要解释一下在数学模式中有哪些尺寸可以使用, 以及 TeX 的选择准则是什么。

在数学模式中可以选四种字体尺寸, 它们的实际尺寸是相对于文档类的基础字体尺寸的:

$\backslash displaystyle$	D	单独公式的标准尺寸
$\backslash textstyle$	T	正文公式的标准尺寸
$\backslash scriptstyle$	S	上下标的标准尺寸
$\backslash scriptscriptstyle$	SS	更低层上下标的标准尺寸

从现在开始我们就采用这里的符号缩写 D , T , S 和 SS 表示相应的尺寸。当切换到数学模式时, 被激活的字体是 D (单独公式) 或 T (正文公式)。它们的差别就在于那些有两种尺寸的符号, 以及上下标是否处理为上下限 (5.3.6 节)。大符号属于 D , 小符号属于 T 。

从这两个基础尺寸出发, 各部分数学要素被设置为相应尺寸。一旦为某个要素选定了一种尺寸, 那么这个尺寸就在这个要素中起作用。

下面的表格说明的是选择规则:

激活 尺寸	分数 上	上、 下	下标 下标
D	T	T	S
T	S	S	S
S	SS	SS	SS
SS	SS	SS	SS

如果被激活的尺寸是 D ，那么就为分数的两部分选择尺寸 T 。也就是说对于分子和分母，被激活的尺寸是 T 。如果它们还包括分数，这时分数两部分的尺寸就是 S 。如果被激活尺寸是 D 或 T ，那么上标和下标 (指数和指标) 的尺寸就是 S ；在上下标中 S 是激活尺寸，它的分数或上下标的尺寸就是 SS 。

TeX 命令 `{ \atop }` 和 `{ \choose }` 被当做分数对待。在 `array` 环境中激活的字体尺寸是 T 。最小的可应用数学字体尺寸是 SS 。一旦已达到了这个尺寸，就不会再小了，因此所有更低层的上标和下标都是 SS 。

从上面的表格很容易看出

`\[a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}} \]`

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}$$

的结果一定是右边那样的形式。

类似于上面这样的数学结构，称为连分数，然而对连分数通常采用如下的排版方法:

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}$$

`\[a_0 + \frac{1}{\displaystyle a_1 + \frac{1}{\displaystyle a_2 + \frac{1}{\displaystyle a_3 + \frac{1}{a_4}}}} \]`

通过在每部分中显式地给出字体尺寸，我们可以确定被激活的尺寸，从而对 TeX 所选择的尺寸进行修改。在上面这个例子中，每个分母的尺寸都选择了 D 。因此后面的分数就以最大尺寸显示出来。最后那个分数中的 `\displaystyle` 命令可以忽略。(为什么呢?)

上面的尺寸选择表格使得我们可以精确地预见到所处部分公式的数学字体尺寸，这样就可以确定是否有必要显式地定义字体尺寸。下面的例子中在右边列出的是没有显式定义数学字体尺寸的结果。

$$\frac{\frac{a}{x-y} + \frac{b}{x+y}}{1 + \frac{a-b}{a+b}}$$

`\[\frac{\displaystyle \frac{a}{x-y} + \frac{b}{x+y}}{\displaystyle 1 + \frac{a-b}{a+b}} \]`

$$\frac{\frac{a}{x-y} + \frac{b}{x+y}}{1 + \frac{a-b}{a+b}}$$

$$e^{-\frac{x_i - x_j}{n^i + n^j}}$$

`\[e^{\textstyle - \frac{x_i - x_j}{n^i + n^j}} \]`

$$e^{-\frac{x_i - x_j}{n^i + n^j}}$$

$$\left[\left(\begin{array}{cc} \left(\begin{array}{c} ab \\ cd \end{array} \right) & \frac{e+f}{g-h} \\ 0 & \left| \begin{array}{c} ij \\ kl \end{array} \right| \end{array} \right) \right]$$

如果在文档中经常需要显示定义尺寸 (例如在 `array` 环境的每个条目中), 我们就可以通过在导言中加入下列指令以避免每次繁琐地输入指令:

```
\newcommand{\D}{\displaystyle}\newcommand{\T}{\textstyle}...
```

用这种方法, 当需要改变尺寸时, 只要简单地输入 `\D` 或 `\T` 等等就可以了。

实际上, 为了方便初学者, 我们在前面所给出的数学字体选择规则只是一种简化后的结果。为了满足细心读者的需要, 我们下面就给出完整的描述。如果读者想知道关于数学公式中字体尺寸选择的所有细节, 不妨耐心地读读下面的内容。

四种数学字体尺寸 D , T , S 和 SS 中每一种都有一个修正版本 D' , T' , S' 和 SS' 。差别就在于 D , T , S 和 SS 的上标 (指数) 要比 D' , T' , S' 和 SS' 的稍高一点点。仔细比较一下分数线上下指数 2 的位置: $\frac{x^2}{x^2}$ 。在其他情形中有撇和无撇的字体尺寸是一样的。下表给出的是真正的选择规则。

激活	分数		上	下
尺寸	上	下	标	标
D	T	T'	S	S'
D'	T'	T'	S'	S'
T	S	S'	S	S'
T'	S'	S'	S'	S'
S, SS	SS	SS'	SS	SS'
S', SS'	SS'	SS'	SS'	SS'

当在分子或者上标中显式地指定字体尺寸时, 选取的是无撇形式; 而在分母和下标中则选取的是有撇的字体尺寸。

5.6.3 括号符号的手工尺寸调整

在附录 A 中列出了 22 个括号符号, 当给它们加上前缀 `\left ... \right` 命令时, 它们会自动根据被包围公式文本的高度调整其自身尺寸。然而, 也可以在括号命令前面放上命令 `\big`, `\Big`, `\bigg` 或 `\Bigg` 来显式选择一个尺寸。

— $\{\} \square \square \langle \rangle \wedge \parallel \uparrow \downarrow \updownarrow$

`\big` $\{\} \square \square \langle \rangle \wedge \parallel \uparrow \downarrow \updownarrow$

`\Big` $\{\} \square \square \langle \rangle \wedge \parallel \uparrow \downarrow \updownarrow$

`\bigg` $\{\} \square \square \langle \rangle \wedge \parallel \uparrow \downarrow \updownarrow$

`\Bigg` $\{\} \square \square \langle \rangle \wedge \parallel \uparrow \downarrow \updownarrow$

与 `\left ... \right` 命令对不同的是, 显式括号尺寸命令并不需要一定包含在公式的一行中。开括号与闭括号可以处在不同的行上。这条规则也同样适用于下面段落中描述的命令。

也存在着另外两组名称为 `\bigl ... \Biggl` 和 `\bigr ... \Biggr` 的括号尺寸命令。这两组命令把后接的括号符号当做开或闭的括号处理。但是在 \LaTeX 中, 这些命令与标准命令之间的差别很小, 因此没有必要把它们区分开。

更进一步的括号尺寸命令是 `\bigm ... \Biggm`, 这些命令把后接的括号符号当做关系运算符处理, 在其前后与相邻公式之间插入较大的水平间距。

<code>\[\big[(a+b) \big (c+d) \big]</code>	$[(a+b) (c+d)]$
<code>\[\bigr[(a+b) \bigr (c+d) \bigr]</code>	$[(a+b) (c+d)]$
<code>\[\bigm[(a+b) \bigm (c+d) \bigm]</code>	$[(a+b) (c+d)]$

5.6.4 数学样式参数

\LaTeX 给下面列出的数学样式参数赋予了标准值, 用户可随时用命令 `\setlength` 按通常的方式改变它们的值。

`\arraycolsep`

array 环境中列间距宽度的一半 (也可见 6.6.2 节)。

`\jot` 在 `eqnarray` 和 `eqnarray*` 环境中插入多行公式两行之间的额外竖直距离。

`\mathindent`

当选择了文档类选项 `fleqn` 时数学公式的缩进量。

`\abovedisplayskip`

当单独公式的左边到左页边界的距离, 小于其前面文本行结尾到左页边界的距离时, 就把这个竖直间距插入到单独公式上方。以后我们称这样的公式为超长公式。

`\belowdisplayskip`

插入在超长单独公式下方的竖直距离。

`\abovedisplayshortskip`

插入在极短单独公式上方的竖直距离。所谓极短公式就是公式的左边在前面文本行结尾的右边。

`\belowdisplayskip`

插入在极短单独公式下方的竖直距离。

`\topsep`

当选择了文档类选项 `fleqn` 时并不使用上述四种距离, 这时要在单独公式的上方和下方插入 `\topsep` (见 4.4.2 节)。

上面除 `\jot` 外的所有参数都应该是弹性长度 (2.5.2 节)。

5.6.5 更进一步的建议

有时候作者会希望得到一些水平或竖直对齐的公式, 但是由于这些公式太复杂, 采用到现在为止描述的方法, 还无法得到令人满意的结果。这时可以考虑把公式放在水平或竖直盒子中, 从而就可以随心所欲地定位。

类似地, 在 `array` 环境中, 可以综合利用显示尺寸声明以及来自于 6.6.1 节和 6.6.2 节中的表格构造和样式要素, 从而实现我们所期望的水平或竖直定位。

例如, 借助于 `\hfill` 命令可以生成下边这样的连分数:

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}$$

$$\backslash[a_0 + \frac{1\hfill}{\displaystyle a_1} + \frac{1\hfill}{\displaystyle a_2} + \frac{1\hfill}{\displaystyle a_3} + \frac{1\hfill}{a_4}]$$

而在 `array` 环境格式域中的 `@{...}` 可以用来插入两列间的水平间距或数学文本, 从而很容易实现如下结构:

$$\begin{array}{ll} \sin 2\alpha = 2 \sin \alpha \cos \alpha, & \cos 2\alpha = \cos^2 \alpha - \sin^2 \alpha, \\ \sin 3\alpha = 3 \sin \alpha - 4 \sin^3 \alpha, & \cos 3\alpha = 3 \cos^3 \alpha - 3 \cos \alpha, \\ \sin 4\alpha = 8 \cos^3 \alpha \sin \alpha - 4 \cos \alpha \sin \alpha, & \cos 4\alpha = 8 \cos^4 \alpha - 8 \cos^2 \alpha + 1. \end{array}$$

5.6.6 有框的单独公式

在 5.4.9 节中我们给出了一种给单独公式加上方框的方法, 即首先把它放在一个适当宽度的 `\parbox` 或 `minipage` 中, 然后再把它们放在 `\fbox` 中以生成方框。但这通常需要经过多次尝试, 才可能找到合理的宽度。

然而, 还有另外一种解决方法, 它利用了 5.6.2 节的数学字体尺寸命令, 因而不需要定义有框单独公式的宽度 (该方法是由德国 Kiel 大学的 Günter Green 提出的):

```
\begin{displaymath}      或      \begin{equation}
\fbbox{$ \displaystyle 公式文本 $}
\end{displaymath}      或      \end{equation}
```

在 111 页上的有框公式示例现在的显示结果为:

$$\boxed{\int_0^\infty f(x) dx \approx \sum_{i=1}^n w_i e^{x_i} f(x_i)} \quad (5.14)$$

生成文本为:

```
\begin{equation}
\fbbox{$ \displaystyle \int^{\infty}_0 f(x)\,, dx \approx \dots $}
\end{equation}
```

由于这里用的是 `equation` 环境, 因此在最右边给公式自动加上了编号, 而方框只是围住了真正的公式。但是如果采用以前的方法, 公式编号也要放在一个盒子中。与 111 页上的例子相比, 现在方框与方程距离更近了。用户可以通过修改参数 `\fbboxsep`(4.7.8 节) 来改变这一距离。类似地, 方框的线粗也可以用 `\fbboxrule` 来设置。另外, 也可以借助于 `amsmath` 命令 `\boxed` 命令达到同样的目的, 例如

```
\begin{equation}
\boxed{\int^{\infty}_0 f(x)\,, dx \approx \dots}
\end{equation}
```

而得到同样的输出, 但后者的输入更简明一些。 `\boxed` 命令也可以用文本参数。

采取同样的方法, 可以利用 `array` 环境, 得到有框的多行公式或方程组:

```
\begin{displaymath} \quad \text{或} \quad \begin{equation}
\fbbox{$ \begin{array}{rcl} \text{公式文本} \end{array} $}
\end{displaymath} \quad \text{或} \quad \end{equation}
```

由于在这一应用中数学字体尺寸命令被相当频繁地使用, 因此我们还是建议采用 120 页上的方法, 重定义命令名称。

5.6.7 补遗

在这一章中列出的数学结构要素以及格式调整 (精调) 方法, 可以满足数学写作方面的大多数需要, 即使应付一些离奇的要求也应是绰绰有余的。在 5.6.5 节中给出的建议更使得所期望的定位和公式构造成为可能, 余下的问题就是如何积累丰富的经验, 充分利用所提供的各种功能了。

如果所需要的某一符号, \LaTeX 并没有提供, 那么我们可以尝试从已有的符号, 利用叠印、提升或降低等操作来构造出来。关于这一点, 请见第 96 页上 `\oiint` 命令的定义。通过利用 `\newcommand` 命令, 可以简化对这种组合符号的使用, 另外为了一般用途, 可以把它的定义存贮在一个文件中。

如果某个构造利用 \LaTeX 是无法达到的, 那么借助于 \TeX 总是可以实现的。然而, 我们要指出的是, 超出 \LaTeX 的任何 \TeX 结构要素都需要对 \TeX 的工作机制有相当的了解, 而大多数用户是做不到这一点的。

为了满足这种要求, 一种更具有足够弹性的比较恰当的设计方法 (但同时也是更乏味的解决方法) 就是利用下一章中关于画图的命令集合。见第 134 页上表示圆弧的符号的定义。

如果作者坚持使用他自己的公式格式, 那么当他把自己的文稿送到科技出版社时, 他自己的格式有可能与国际标准相差太远, 编辑们就可能把它重新设成与 \LaTeX 类似的

形式。

另一方面，对公式编号样式改变的要求通常就是如何对齐。在 7.3.4 节中给出了一个例子，说明如何做到这一点。可以把它做为一个样板。

第六章 图 表

利用 \LaTeX 可以生成简单的图形。由文本、各种斜率的直线、箭头、圆、卵形线和线段等组成了构造图形的模块，用户可以把这些模块放在任何所希望的地方。如果文章中只是需要生成简单的示意图，由 \LaTeX 所提供的画图功能一般来说能够满足要求。在 8.11 节中会讨论如何插入用其他软件生成的复杂图形。我们在这一章还同时讲解了浮动对象的安排问题。

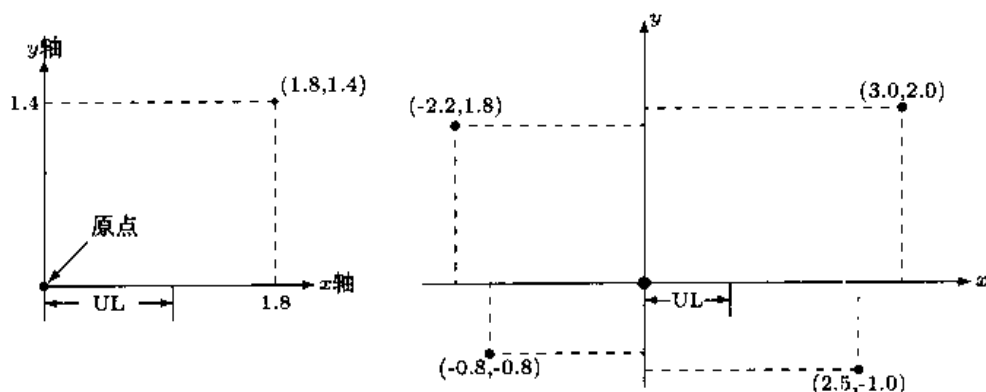
6.1 图形的尺寸和位置

在安置构造模块之前，必须先为图形建立一个定位系统（即坐标系），而坐标系由参考点（也称为原点）和两条互相垂直的坐标轴组成，坐标轴上要有单位长度以确定位置。一般就把原点放在图形的左下角，坐标轴就是下底边与左边界。这两条边分别被称为 x 轴（底边）和 y 轴（左边）。

单位长度的选择是用如下命令进行的：

```
\setlength{\unitlength}{长度}
```

一旦定义了单位长度，图形中每点都唯一被两个数确定：第一个是沿 x 轴的长度，第二个是沿 y 轴的长度。



在上图左边的那个例子中，单位长度用 `\setlength{\unitlength}{1.5cm}`，设置成 1.5cm，这样点 (1.8, 1.4) 的位置就是在原点右边 1.8 倍单位长度 (= 2.7cm)，然后向上移动 1.4 倍单位长度 (= 2.1cm)。

由于参考点通常在图形的左下角，因此所有点都应该在它的右上方向，也就是说，坐标值一般都是正数。然而，坐标也可以取负值。负的 x 值（坐标中第一个数为负数）定义的点在原点的左边，而负的 y 值（坐标对中第二个数为负数）定义的点在原点的下边。

上图右边的例子说明了这种情形。这里的单位长度为 1cm，因此坐标值就是两个方向上离原点以 cm 为单位的距离。

单位长度通常就取为方便的尺寸, 例如 1cm, 1mm 或 1in, 然后相应地建立我们的图形。当整幅图形完成后, 只需修改单位长度的定义, 就可以放缩图形。一幅图在原始设计时 `\unitlength` 取值 1cm, 那么就可以通过把单位长度重新定义为 1.2cm 来放大到原来的 1.2 倍。

6.2 图形环境

图形是用 `picture` 环境构造的, 其语法为

```
\begin{picture}(x 尺寸, y 尺寸)
    画图命令
\end{picture}
```

这里的 $(x \text{ 尺寸}, y \text{ 尺寸})$ 是一组数, 它定义了图形在 x 方向 (水平) 和 y 方向 (竖直) 的尺寸 (范围)。注意这对数需要用小括号围起来! 单位长度就是在此之前所定义的 `\unitlength`。

```
\setlength{\unitlength}{1.5cm}
\begin{picture}(4,5) ... .. \end{picture}
```

的结果为一幅 4 个单位长度宽, 5 个单位长度高的图形。而由于已把单位长度设为 1.5cm, 因此图形的实际尺寸为宽 6cm, 高 7.5cm。

而 画图命令 就是下面将要介绍的用以生成和定位各个图形要素的命令。这些命令再加上字体样式和尺寸声明 (4.1 节) 以及线的粗细命令 `\thicklines` 与 `\thinline` 就是可以位于 `picture` 环境中的所有命令。而线的粗细命令可以用来确定当前直线的粗细, 我们可以按自己的需要在两者之间来回切换。刚开始时激活的是细线。

参数 `\unitlength` 的值一定不能在 `picture` 环境内改变, 因为对它而言, 单位长度必须维持不变。当然可以在两幅图形之间改变它的值。

如果 `\unitlength` 定义随同 `picture` 环境一起被包围在另一个类似于 `\begin{center} ... \end{center}` 这样的环境内, 那么 `\unitlength` 值的作用持续到这个环境结束。前面没有定义 `\unitlength` 的 `picture` 环境, 采用的是单位长度的标准值 1pt。

6.3 定位命令

图形中的元素是用 `\put` 和 `\multiput` 这两条命令来创建和定位的, 它们的语法为:

```
\put(x坐标, y坐标){ 图形元素 }
\multiput(x坐标, y坐标)(x增量, y增量){ 数 }{ 图形元素 }
```

这里的 图形元素 就是下一节要讲述的画图基本命令。参数值 $(x \text{ 坐标}, y \text{ 坐标})$ 是安置坐标, 规定了元素在图形坐标系中的位置, 它是以 `\unitlength` 为单位。如果单位长度为 1cm, 那么 $(2.5, 3.6)$ 就意味着元素定位在相对于图形左下角向右 2.5cm, 向上 3.6cm 的地方。

命令 `\multiput` 是把同样的图形元素生成数次, 每次移动 (x 增量, y 增量)。因此元素被重复地画在

(x 坐标, y 坐标), (x 坐标 + x 增量, y 坐标 + y 增量),
 (x 坐标 + $2x$ 增量, y 坐标 + $2y$ 增量), ..., 一直到
 (x 坐标 + $[数-1]x$ 增量, y 坐标 + $[数-1]y$ 增量)

每画一次, (x 坐标, y 坐标) 都要增加 (x 增量, y 增量)。这里被增加的量可以是正数, 也可以是负数。

因此 `\multiput(2.5,3.6)(0.5,-0.6){5}{图形元素}` 就会画五次图形元素, 第一个图形元素的位置是 (2.5,3.6), 然后分别是 (3.0,3.0), (3.5,2.4), (4.0,1.8), 最后一个位于 (4.5,1.2)。

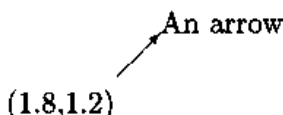
注意坐标和增量数对都是放在小括号 (,) 内, 其中的两个数是用逗号分开的。而数和图形元素项则是像通常那样放在大括号内。

警告: 由于这里是用逗号分开两个数, 因此不能再用它表示小数点。对于坐标项, 小数点必须是 (半角) 句号, 而不能是逗号。

6.4 基本画图命令

6.4.1 图形中的文本

所有图形元素中最简单的就是把一段文本定位在图形中给定的位置。为此只要把文本放在 `\put` 或 `\multiput` 命令中的图形元素所在地方就可以了。

 (1.8,1.2) 箭头所指的位置为 (1.8,1.2)。
 利用命令 `\put(1.8,1.2){An arrow}` 就会插入文本 'An arrow', 其左下角就是命令中所指定的位置。

做为图形元素的文本也可以包含在一个 `\parbox` 或 `minipage` 环境中, 这时在 `\put` 命令中坐标项的参考点与竖直盒子的定位参数值有关:

<code>\parbox[b]{...}{...}</code>	<code>\parbox{32cm}{...}</code>	<code>\parbox[t]{...}{...}</code>
Reference point is the lower left corner of the last line in the parbox	For a standard parbox, the reference point is the vertical center of the left edge	Reference point is the lower left corner of the top line in the parbox

6.4.2 图形中的盒子

在 `picture` 环境中也可用 `\framebox`, `\makebox` 和 `\savebox`(4.7.1 节) 等盒子命令, 但是其语法已做了推广。而且, 还有另一个盒子命令 `\dashbox`:

```
\makebox( $x$ 尺寸,  $y$ 尺寸)[位置]{文本}
\framebox( $x$ 尺寸,  $y$ 尺寸)[位置]{text}
\dashbox{虚线尺寸}( $x$ 尺寸,  $y$ 尺寸)[位置]{文本}
```

(x 尺寸, y 尺寸) 数定义了矩形的宽度和高度, 它们以 `\unitlength` 为单位。定位参数值位置 定义了文本在盒子中的位置。它可以取如下值:

- [t] top- 输入文本水平居中地位于盒子顶边的下面。
- [b] bottom- 输入文本水平居中地位于盒子底边的上面。
- [l] left- 输入文本竖直居中地位于盒子的左边。
- [r] right- 输入文本竖直居中地位于盒子的右边。
- [s] stretch- 输入文本竖直居中, 但要水平伸展以充满整个盒子。

如果没有可省参数 位置, 那么输入文本是水平竖直地居中放置在盒子中。

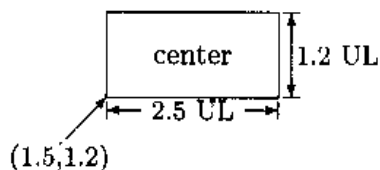
也可以如下把这些参数中的两个组合起来使用:

- [tl] top left- 输入文本位于左上角。
- [tr] top right- 输入文本位于右上角。
- [bl] bottom left- 输入文本位于左下角。
- [br] bottom right- 输入文本位于右下角。

这里值的顺序是无关紧要的, 例如 `tl` 与 `lt` 的效果相同。

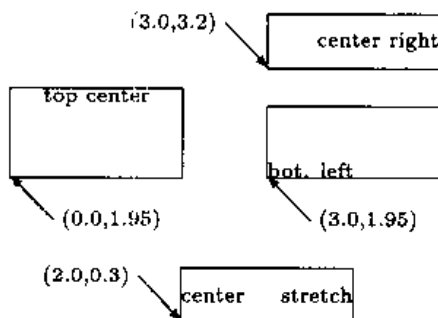
这些命令可以用在 `\put` 和 `\multiput` 命令的 图形元素 中。在放置盒子时, 其左下角就对应于放置命令中的坐标对。

`\put(1.5,1.2){\framebox(2.5,1.2){center}}`



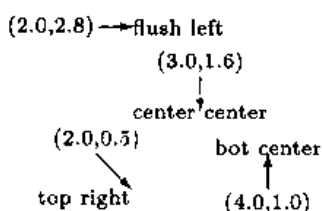
箭头所指位置为 (1.5,1.2), 这就是宽 2.5 单位, 高 1.2 单位的矩形左下角所在的地方。文本 'center' 水平和竖直居中。UL = 0.8cm。

下面这个例子更好地说明了文本定位参数值的作用 (UL = 1cm):



```
\put(0.0,1.95){\framebox(2,1.0)
[t]{top center}}
\put(3.0,1.95){\framebox(2,0.8)
[lb]{bot. left}}
\put(3.0,3.2){\framebox(2,0.6)
[r]{center right}}
\put(2.0,0.3){\framebox(2,0.6)
[s]{center\hfill stretch}}
```

图形元素 `\makebox` 同 `\framebox` 命令完全一样, 只是它不画出矩形框。对这条命令而言, 经常把范围坐标对取为 (0,0), 这样可以把文本放在所希望的地方。(关于零宽度盒子对被包围文本的效果请见 4.7.1 节。)

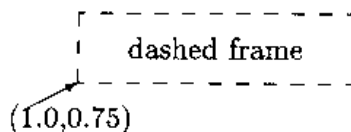


```
\put(3,1.6){\makebox(0,0){center center}}
\put(2,0.5){\makebox(0,0)[tr]{top right}}
\put(4,1.0){\makebox(0,0)[b]{bot center}}
\put(2,2.8){\makebox(0,0)[l]{flush left}}
[lb] 组合定位的文本就与不用盒子, 直接把文本做为 图形
元素 时的效果一样, 见 6.4.1 节。
```

图形元素 `\dashbox` 也生成有框盒子, 不过其框线为虚线。参数值 虚线尺寸 用来定义短线长度。

```
\put(1.0,0.75){\dashbox{0.2}(4,1){dashed frame}}
```

当盒子的宽度和高度都是短线长度的倍数时, 虚线框要好看一些。



即使上面这些图形盒子命令中, 也可以把文本放在竖盒子 (`\parbox` 或 `minipage`) 中。由于竖盒子自身具有可省的定位参数值 `b` 或 `t`, 它一定不能与图形盒子的定位参数值冲突, 因此要遵守下面的规则:

如果图形盒子中包含了定位参数值 `b` 或 `t`, 那么被包围的竖盒子中也必须有相同的定位参数值。如果图形盒子中没有定位参数值, 或者只是 `r` 或 `l`, 那么竖盒子必须是标准 (无参数值) 形式。

图形盒子中的定位参数值对于被包围竖盒子的作用同它对一行文本的作用一样, 都是把它们当做一个整体对待。

6.4.3 直线

在 `picture` 环境中, \LaTeX 可以绘制任意长度的水平、竖直以及有限倾角的直线。这个图形元素的语法是:

```
\line( $\Delta x, \Delta y$ ){长度}
```

对于水平线和竖直线, 长度 定义了以单位长度为单位的线长。对于其他倾角的直线, 它的意义就有点儿复杂, 稍后加以解释。直线开始于由 `\put` 或 `\multiput` 命令中给出的安置坐标确定的点。

```
\thicklines
```

```
\put(0,0){\line(1,0){6}}
```

```
\put(0,0){\line(0,1){1}}
```

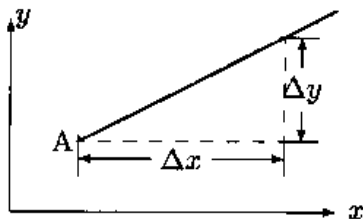
```
\put(6,0){\line(0,1){0.5}}
```



绘制直线时, 直线的倾角是由斜率对 $(\Delta x, \Delta y)$ 给定的。斜率对 $(1,0)$ 中 $\Delta x = 1$, $\Delta y = 0$, 这样生成水平线, 而 $(0,1)$ 生成的则是竖直线。上面的例子说明了这一点。一般地 $(\Delta x, \Delta y)$ 具有下面的含义:

从直线上的 A 点开始, 沿着 x 方向 (水平) 走 Δx 距离, 那么 Δy 就是沿 y 方向 (竖直) 移动的距离, 从而可以重新回到直线上。

通过定义斜率对 $(\Delta x, \Delta y)$, 绘制出来的直线倾角应该满足上面的条件。



前面已经说过, 只可以绘制有效倾角的直线。这是因为 Δx 和 Δy 的取值要遵从下面的规则:

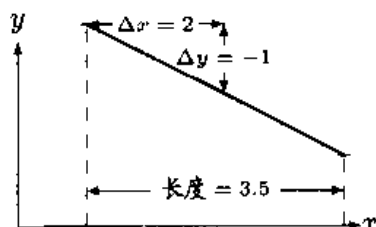
1. 数值必须是整数 (负数或正数均可)。

2. 只可以取值 $0, 1, \dots, 6$ 。

3. 在数对中的两数不能有公因子。

因此类似 $(3.5, 1.2)$ (规则 1) 和 $(7, 0)$ (规则 2) 这样的数对是不允许出现的。同样 $(2, 2)$ 和 $(3, 6)$ 不符合规则 3, 因为前一对中的两个数可以都被 2 整除, 而后一对中的两数可以都被 3 整除。可以用 $(1, 1)$ 和 $(1, 2)$ 来得到同样的倾角。因此这里一共有 25 种可接受的斜率对, 其中 $(1, 0)$ 和 $(0, 1)$ 分别相应于水平线和竖直线。不妨把所有的可能写出来, 验证这一总数。

另外, 在斜率对中的数值可以是正数, 也可以是负数, 例如 $(0, -1)$ 和 $(-2, -5)$ 也是允许的。在上面图示中, 负的 Δx 意味着向左移动, 而负的 Δy 意味着向下移动。因此 `\put(2,3){\line(0,-1){2.5}}` 的结果是一条开始于 $(2, 3)$ 的直线, 其竖直向下伸展长达 2.5 单位。



对于有倾角的直线, 参数值 长度 定义的是沿 x 轴的投影长度。这一点可以借助于左边的图示更清楚地看出来。

`\put(1.0,2.75){\line(2,-1){3.5}}`

如果从两个端点竖直向下画虚线, 那么在这两虚线之间的 x 轴部分就是直线在 x 轴上的投影。

倾斜直线的长度必须不能短于 10pt (即 3.5mm), 否则不会生成任何结果。但是如果包含了 `pict2e` 宏包 (6.5.6 节), 就不会有这种限制。

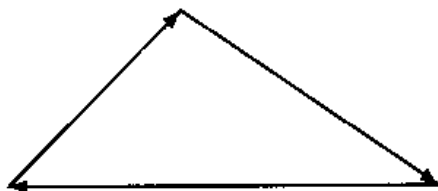
6.4.4 箭头

箭头图形元素是用下面的命令生成的:

`\vector(Δx, Δy){ 长度 }`

其作用方式同 `\line` 命令完全一样, 而且参数值及其所要遵从的规则也是相同的。这条命令从由 `\put` 或 `\multiput` 命令定义的位置开始画一条直线, 然后在终点处画上箭头。

同直线一样, 箭头的长度也不能少于 10pt 或 3.5mm。规则 1-3 也同样适用于 Δx 和 Δy , 而且更进一步, 要求可以取的值只能是 0, 1, 2, 3, 4。这样当不考虑正负号时, 只能画 13 种不同倾角的箭头。



```
\begin{picture}(5,2)\thicklines
\put(5,0){\vector(-1,0){5}}
\put(0,0){\vector(1,1){2}}
\put(2,2){\vector(3,-2){3}}
\end{picture}
```

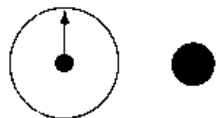
6.4.5 圆

圆是用下面的命令得到的:

`\circle{ 直径 }`

`\circle*{ 直径 }`

利用 `*`- 形式的命令, 可以画出内部被填充了的实心圆, 而标准形式画出的则只是轮廓线。由于只能画特定尺寸的圆, 因此 \LaTeX 会选择与指定直径最接近的圆。(6.5.6 节介绍的 `pict2e` 宏包可以绘制任意尺寸的圆。)



```
\begin{picture}(3,1.6)
  \put(1,1){\circle*{0.2}}
  \put(1,1){\circle{1.2}}
  \put(1,1){\vector(0,1){0.6}}
  \put(2.5,1){\circle*{0.5}}
\end{picture}
```

在相应的 `\put` 命令中的安置位置对应于圆心。

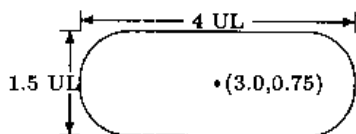
6.4.6 卵形线与圆角

我们这里所说的卵形线指的是一种矩形, 其顶角用四分之一圆周代替; 这里对直径的选取是使所有边光滑拼接的最大值。生成卵形线的命令是

`\oval(x尺寸, y尺寸)[部分]`

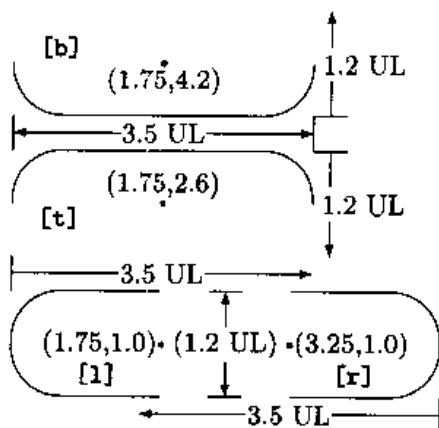
在相应 `\put` 命令中的安置坐标对应于卵形线的中心。

`\put(3.0,0.75){\oval(4.0,1.5)}`



这里我们取 x 尺寸 = 4.0UL, y 尺寸 = 1.5UL, 而单位长度 UL 已选择为 0.8cm。卵形线的中心就是 `\put` 命令中的安置坐标 (3.0, 0.75)。

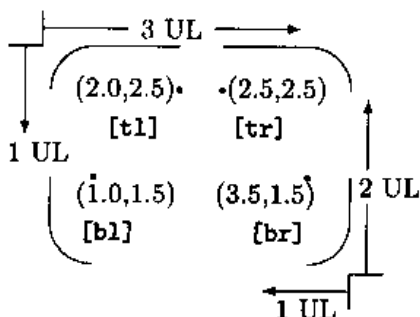
可省参数 `部分` 的取值可以为 `t`, `b`, `l` 或者 `r`, 以生成半个卵形线。



```
\put(1.75,4.2){\oval(3.5,1.2)[b]}
\put(1.75,2.6){\oval(3.5,1.2)[t]}
\put(1.75,1.0){\oval(3.5,1.2)[l]}
\put(3.25,1.0){\oval(3.5,1.2)[r]}
```

虽然这里只是画出卵形线的一半, 但其宽度与高度与整个都要画出来时是一样的。类似地, 在相应 `\put` 命令中的安置坐标仍然对应于完整卵形线的中心。(这里的单位长度为 $\text{UL} = 1\text{ cm}$ 。)

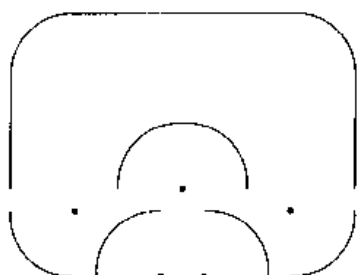
参数值 `部分` 也可以是四种组合 `tl`, `tr`, `bl` 或 `br` 中的一种, 以生成四分之一的卵形线。这里两字母的顺序是无关紧要的, 因此也可以用 `lt`, `rt`, `lb` 或 `rb`。



```
\put(2.0,2.5){\oval(3.0,1.0)[tl]}
\put(2.5,2.5){\oval(3.0,1.0)[tr]}
\put(1.0,1.5){\oval(1.0,2.0)[bl]}
\put(3.5,1.5){\oval(1.0,2.0)[br]}
```

同样这里的尺寸定义仍旧参照完整卵形线而进行, 即使画出来的只是一部分, 在 `\put` 命令中的安置坐标也是指的整个卵形线的中心。

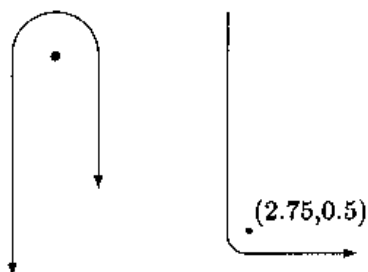
通过把卵形线中宽度与高度取成相等的值,可以得到四分之一或者半个圆周,但正如前面对圆的限制一样, \LaTeX 为给定尺寸选择一个最近的尺寸。



```
\put(2.0,1.0){\oval(4.0,4.0)[t]}
\put(2.0,1.0){\oval(1.5,1.5)[t]}

\put(0.75,0.75){\oval(1.5,1.5)[bl]}
\put(1.75,0.0){\oval(1.5,1.5)[tl]}
\put(2.25,0.0){\oval(1.5,1.5)[tr]}
\put(3.25,0.75){\oval(1.5,1.5)[br]}
```

部分卵形线也可以与其他图形要素组合。当然这时可能需要对 `\put` 命令中的安置坐标进行相当仔细的考虑,以准确定位。



```
\put(0.5,2.5){\oval(1.0,1.0)[t]}
\put(0.0,2.5){\vector(0,-1){2.5}}
\put(1.0,2.5){\vector(0,-1){1.5}}
\put(0.5,2.5){\circle*{0.1}}

\put(2.5,0.5){\line(0,1){2.5}}
\put(2.75,0.5){\oval(0.5,0.5)[bl]}
\put(2.75,0.25){\vector(1,0){1.25}}
```

在上面所有例子中,卵形线的中心都用一个点标记出来,以说明其位置。通常画图时并不需要这样做。

6.4.7 竖直堆积文本

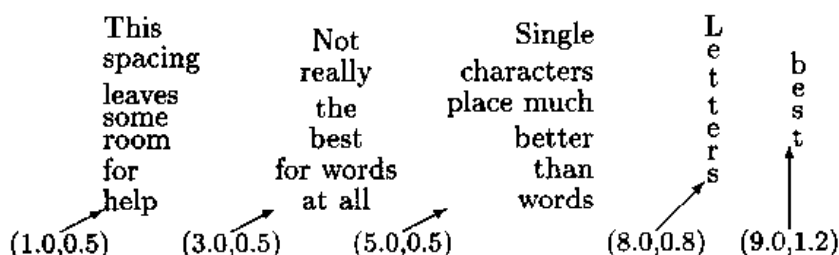
有时在演示图中需要沿竖直方向书写文本,如本段右边文本所示。这可以用如下命令来做到

`\shortstack[位置]{列}`

位置 参数值可以取 `l`, `r` 或 `c`。标准值是 `c`, 表示居中。该命令类似于只有一列的 `tabular` 环境。列 表示输入文本,行与行之间用 `\\` 分开。

y
-
a
x
i
s

`\shortstack` 命令经常用来实现把单个字母上下安置,即使是长文本也可以竖直堆积。行与行之间用尽可能小的间距分开。也就是说没有向下或向上突出字母(如 `h` 和 `y`)的行与相邻行之间的距离,大于有这样字母的行与相邻行之间的距离。



相应 `\put` 命令中的安置坐标对应于想像中的包含竖直堆积文本的盒子的左下角。上面例子中第一个的文本是左对齐的，第二个是居中，第三个是右对齐。最右面的两个是居中的，它们是用如下输入生成的：

```
\put(1.0,0.5){\shortstack[l]{This\\spacing\\leaves\\some\\ ...}}
\put(3.0,0.5){\shortstack{Not\\really\\the\\best\\ ...}}
\put(5.0,0.5){\shortstack[r]{Single\\characters\\ ...}}
\put(8.0,0.8){\shortstack{L\\e\\t\\t\\e\\r\\s}}
\put(9.0,1.2){\shortstack{b\\e\\s\\t}}
```

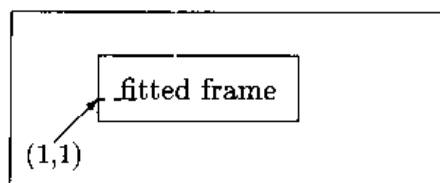
`\shortstack` 命令也可以用在 `picture` 环境外面的普通文本中。其中一个应用就是页边注。见 4.10.6 节。

6.4.8 有框文本

`\framebox` 命令生成一个具有预定义尺寸的有框盒子，其中的文本可以放在各种不同的地方 (6.4.2 节)。在文本模式中，我们是用 `\fbox` 命令围绕文本画一个方框，而且方框与被包围文本匹配得相当好 (4.7.1 节)。这条命令也可以用在 `picture` 环境中。

在盒子的框线与被包围文本之间的距离是由参数 `\fboxsep` 所确定的。利用 `\put` 命令放置一个 `\fbox` 时, 结果会出乎我们的意料, 请看下图:

```
\begin{picture}(5,2)
\setlength{\fboxsep}{0.25cm}
\put(0,0){\framebox(5,2){}}
\put(1,1){\fbox{fitted frame}}
\end{picture}
```



注意上面例子中定位参数的效果。

在演示图中通常不希望框中出现多余的空白，特别当方框包围的是一幅图，而不是文本时更是如此。在这种情况下，可以使用命令

`\frame{ 图形元素 }`

\put 命令中的安置坐标与通常一样, 相应于左下角。

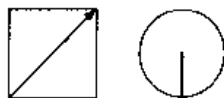
```
\put(0.0,0.5){\frame{TEXT}}
```

```
\put(1.5,0.0){\frame{\shortstack{W\\O\\R\\D}}}
```

TEXT	WORD
1. The first of the two main types of text is the <u>descriptive</u> text. This type of text is used to describe a person, a place, a thing, or an event. It is often found in textbooks, travel guides, and news reports.	1. The first of the two main types of text is the <u>descriptive</u> text. This type of text is used to describe a person, a place, a thing, or an event. It is often found in textbooks, travel guides, and news reports.
2. The second type of text is the <u>argumentative</u> text. This type of text is used to present a point of view on a particular issue and to persuade the reader to accept that point of view. It is often found in essays, editorials, and opinion columns.	2. The second type of text is the <u>argumentative</u> text. This type of text is used to present a point of view on a particular issue and to persuade the reader to accept that point of view. It is often found in essays, editorials, and opinion columns.
3. The third type of text is the <u>expository</u> text. This type of text is used to explain a process, a concept, or a problem. It is often found in textbooks, technical manuals, and scientific articles.	3. The third type of text is the <u>expository</u> text. This type of text is used to explain a process, a concept, or a problem. It is often found in textbooks, technical manuals, and scientific articles.
4. The fourth type of text is the <u>narrative</u> text. This type of text is used to tell a story or to describe a sequence of events. It is often found in novels, short stories, and news reports.	4. The fourth type of text is the <u>narrative</u> text. This type of text is used to tell a story or to describe a sequence of events. It is often found in novels, short stories, and news reports.
5. The fifth type of text is the <u>persuasive</u> text. This type of text is used to persuade the reader to take a particular action or to believe in a particular idea. It is often found in advertisements, political speeches, and opinion columns.	5. The fifth type of text is the <u>persuasive</u> text. This type of text is used to persuade the reader to take a particular action or to believe in a particular idea. It is often found in advertisements, political speeches, and opinion columns.
6. The sixth type of text is the <u>informative</u> text. This type of text is used to provide information about a particular topic. It is often found in textbooks, encyclopedias, and news reports.	6. The sixth type of text is the <u>informative</u> text. This type of text is used to provide information about a particular topic. It is often found in textbooks, encyclopedias, and news reports.
7. The seventh type of text is the <u>analytical</u> text. This type of text is used to analyze a problem or a situation and to provide a solution. It is often found in textbooks, technical manuals, and scientific articles.	7. The seventh type of text is the <u>analytical</u> text. This type of text is used to analyze a problem or a situation and to provide a solution. It is often found in textbooks, technical manuals, and scientific articles.
8. The eighth type of text is the <u>evaluative</u> text. This type of text is used to evaluate a person, a place, a thing, or an event. It is often found in textbooks, travel guides, and news reports.	8. The eighth type of text is the <u>evaluative</u> text. This type of text is used to evaluate a person, a place, a thing, or an event. It is often found in textbooks, travel guides, and news reports.
9. The ninth type of text is the <u>comparative</u> text. This type of text is used to compare two or more things and to highlight their similarities and differences. It is often found in textbooks, travel guides, and news reports.	9. The ninth type of text is the <u>comparative</u> text. This type of text is used to compare two or more things and to highlight their similarities and differences. It is often found in textbooks, travel guides, and news reports.
10. The tenth type of text is the <u>reflective</u> text. This type of text is used to reflect on a person's own experiences and to provide a personal perspective on a particular issue. It is often found in essays, editorials, and opinion columns.	10. The tenth type of text is the <u>reflective</u> text. This type of text is used to reflect on a person's own experiences and to provide a personal perspective on a particular issue. It is often found in essays, editorials, and opinion columns.

`\frame` 命令中的内容并不一定只是文本，也可以是前面给出的图形元素。然而，在很多情形中输出可能有点儿不对头。

```
\put(0,0){\frame{\vector(1,1){1.0}}}  
\put(2,0){\frame{\circle{1.0}}}
```



第一条命令得到正确的结果，而第二条命令却失败了。对此情况，我们可以尝试把图形对象放在适当尺寸 `\makebox` 中，然后把它作在 `\frame` 命令的参数值。然而，用 `\framebox` 命令代替 `\frame{\makebox...}` 会更合理一些。

6.4.9 曲线

在 `picture` 环境中也可以用下面的命令画曲线:

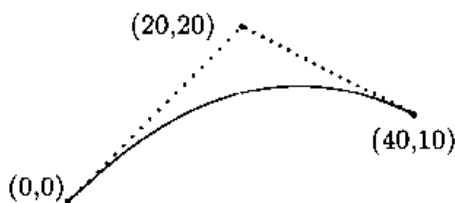
```
\bezier{ 数 }(x_1,y_1)(x_2,y_2)(x_3,y_3)
```

```
\qbezier[ 数 ](x_1,y_1)(x_2,y_2)(x_3,y_3)
```

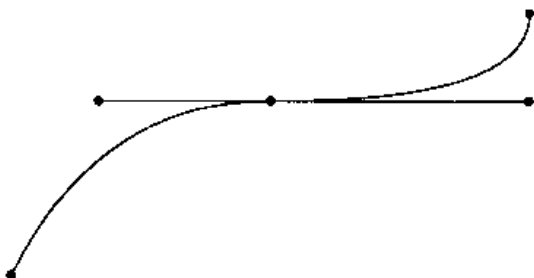
它会画出一条以 (x_1, y_1) , (x_2, y_2) 和 (x_3, y_3) 为控制点的二次 Bézier 曲线, 关于 Bézier 曲线的详情, 请见 [3]。曲线实际上是用 数+1 个点画出来的。 `\bezier` 和 `\qbezier` 命令之间的唯一区别在于, 对后者而言, 数 是一个可省参数值; 如果省略了它, 那么就会计算生成一条光滑曲线所需要的点数。之所以还保留 `\bezier` 命令, 就是为了与原来的 L^AT_EX 2.09 文档兼容。

Bézier 控制点的意义可以从右面的例子看出来。输入文本为

```
\begin{picture}(40,20)
\qbezier(0,0)(20,20)(40,10)
\end{picture}
```



这条曲线是从 $(0,0)$ 到 $(40,10)$, 在端点处的切线 (虚线) 就经过 Bézier 控制点 $(20,20)$ 。上面例子中的虚线并不是 `\qbezier` 函数绘制出来的, 只是为了说明上述含义。利用 Bézier 曲线的性质, 我们可以很容易地拼接出光滑的曲线。为了保证曲线的光滑性, 我们需要两段曲线在一个端点处重合, 前一段曲线的后两个控制点与后一段曲线的第二个控制点共线。例如



```
\begin{picture}(60,30)
\qbezier(0,0)(10,20)(30,20)
\qbezier(30,20)(60,20)(60,30)
\end{picture}
```

通过指定命令中的 数 参数, 可以绘制出点划曲线, 但是要得到比较好的效果, 可能需要试验几次。对于给定的 数, 表示只画出曲线上的 数+1 个点。例如,

```
\begin{picture}(40,20)
\qbezier[10](0,0)(20,20)(40,10)
\end{picture}
```

注意: `\bezier` 命令并不是积成在 L^AT_EX 2.09 中的部分, 它是包含在一个叫 `bezier.sty` 的文件中, 必须把 `bezier` 列在 `\documentstyle` 的选项中, 这样才能读取它。在 L^AT_EX 2_ε 中为了保证兼容性, 提供了同名文本, 但其内容是空的。

有趣的是, 利用 `\qbezier` 命令, 我们可以差强人意地弥补 L^AT_EX 在数学符号上的一个小缺陷。在初等几何中, 经常可以见到, 为了表示圆弧, 在几个大写字母上方放一段圆弧, 例如 \widehat{AOB} 表示圆心为 O , 端点分别为 A 与 B 的一段圆弧。为此, 我们需要进行如下定义:


```

\makeatletter
\newlength{\arclength}
\newcommand{\arc}[1]{\settowidth{\arclength}{#1}%
                    \setlength{\unitlength}{0.01\arclength}%
                    \providecommand{\@arc}{}%
                    \renewcommand{\@arc}{%
                        \begin{picture}(100,30)%
                            \qbezier(0,0)(50,35)(100,0)%
                        \end{picture}}%
                    \stackrel{\@arc}{#1}}
\makeatother

```

这样输入 `\arc{ABC}` 就可以得到 \widehat{AOB} .

6.5 其他图形命令与示例

6.5.1 直线粗细

对于图形元素 `\circle`、`\oval`、`\vector` 和斜线, 存在两种可选择的线宽。可以用

`\thicklines` 或 `\thinlines`

来选择粗线或细线。这两条命令的作用直到有相反命令为止, 或者其所在环境结束。刚开始时 `\thinlines` 起作用。

水平或竖直直线的线宽可以用下面的声明来定义成所希望的任意尺寸:

`\linethickness{粗细}`

参数值 `粗细` 是正的长度定义。利用 `\linethickness{1.5mm}` 可以使得所有后面的水平或竖直线宽度为 1.5mm. 如果包含了 `pict2e` 宏包 (6.5.6 节), 这个线宽定义也同样适用于斜线。

由于方框就是用竖直线和水平线构成的, 因此 `\linethickness` 的定义对 `\framebox` 和 `\dashbox` 也同样有作用。

6.5.2 嵌套图形

在 `\put` 或 `\multiput` 命令中的图形元素也可以是另一个 `picture` 环境。这种多重嵌套图形的语法是:

```

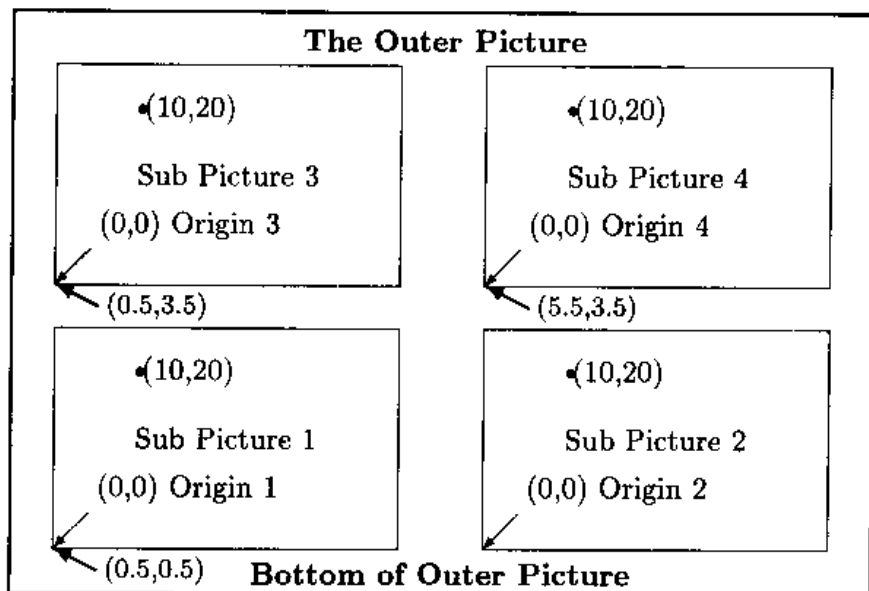
\put(x坐标, y坐标){\setlength{\unitlength}{单位长度}
                    \begin{picture}{x尺寸, y尺寸}
                        ... 子图形 ... \end{picture} }

```

在内部图形环境中的安置坐标是相对于它自己的原点的, 即其左下角; 而原点就对应于外面 `picture` 环境中 `\put` 命令的安置坐标。可以给内部 `picture` 环境选择不同的

`\unitlength` (单位长度) 值; 然而, 如果没有定义新的值, 那么它的单位长度就与外面 `picture` 环境的单位长度相同。

```
\begin{picture}(10.0,6.6)
\thicklines\put(0,0){\framebox(10.0,6.6){}}
\put(5.0,6.3){\makebox(0,0){\bfseries The Outer Picture}}
\thinlines
\put(.5,.5){\setlength{\unitlength}{1mm}}
\begin{picture}(50,25)
\put(0,0){\framebox(40,25){Sub Picture 1}}
\put(10,20){\circle*{1}} \put(10,20){\makebox(0,0)[l]{(10,20)}}
\put(4,4){\vector(-1,-1){4}}
\put(5,5){\makebox(0,0)[lb]{(0,0) Origin 1}} \end{picture}}
\put(5.5,0.5){ ... Sub Picture 2 ... }
\put(0.5,3.5){ ... Sub Picture 3 ... }
\put(5.5,3.5){ ... Sub Picture 4 ... }
\put(5,0.1){\makebox(0,0)[b]
{\bfseries Bottom of Outer Picture}}
\end{picture}
```



在上面的例子中, 最外层图形的单位长度设为 $UL=1\text{ cm}$, 该图形宽 10 cm , 高 6.6 cm . 它的图形要素中有一个粗线的方框, 这个方框的尺寸与图形的尺寸一样, 其他的元素有: 两块文本 'The Outer Picture' 和 'The Bottom of the Outer Picture', 四幅小图形, 在每一幅小图形中, 单位长度为 $UL=1\text{ mm}$, 图形大小为 $40 \times 25\text{ mm}$. 在每个小图形中的对象相对于它们自己的原点定位. 上例中每个小图形中的 \bullet 符号都具有相同的坐标 (10,20)。

通过使用嵌套图形，可以简化逻辑上彼此相关的特定对象之间的相互定位，从而减少了定位时可能出现的错误。尤其是在调用 `\put` 命令时把 `\unitlength` 命令设成了不同的值，更需要这种功能。

6.5.3 存贮部分图形

当我们要用 \LaTeX 的图形命令构造复杂插图时，也许会遇到这样的情形。我们首先精心构造了一个稍有些复杂的小块图形，然后我们想把它复制到别的地方，甚至是另一个图形环境中。如果重复输入构造命令，那么就要进行仔细的重定位，一般情况下，所有 `\put` 命令中的位置参数都要改变。为此，我们可以利用 \LaTeX 的子图以及与子图有关的命令，来简化上述操作。可以把图形中的一组图形元素用特定的名称保存成子图，然后就能随时调调用整组命令，而不再需要逐条调用。

首先必须为每个子图起一个名称，所用命令为

```
\newsavebox{\子图名称}
```

这样就会创建一个名叫 `\子图名称` 的用于保存图形的盒子。然后，用下面的命令保存子图：

```
\savebox{\子图名称}(x尺寸,y尺寸)[位置]{子图}
```

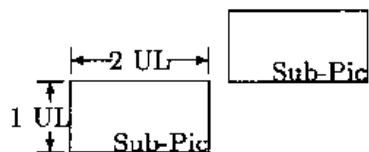
这里的参数值 (x -尺寸, y -尺寸) 与 `位置` 同 6.4.2 节 `\makebox` 中的含义一样。

如果子图只是一小块文本，那么这条命令就与 `\makebox` 命令几乎完全一样，只是文本不显示出来，而是存贮到 `\子图名称` 中。可以用下面这条命令把子图当做一个图形元素安置在主图内的任何地方。

```
\usebox{\子图名称}
```

例如，

```
\newsavebox{\sub}
\savebox{\sub}(2,1)[br]{\small Sub-Pic}
....
\put(0.7,0.0){\frame{\usebox{\sub}}}
\put(3.0,1.0){\frame{\usebox{\sub}}}
```



这个例子看起来并不怎样，因为如果不用 `\savebox` 和 `\usebox` 命令，而把 `\framebox` 和 `\multiput` 命令结合起来使用，就可以更简单地得到同样结果。然而，用子图命令的主要优势并不是安置多重文本，而是表现在安置更复杂子图组合的情形中。

这里要指出的是，`\savebox` 命令可以在 `picture` 环境外面调用，甚至可以把它放在导言中。这样该子图就可以在整篇文档的所有 `picture` 环境中使用。然而，如果 `\savebox` 是定义在一个环境中，那么其中的值当环境结束时也就没有了。

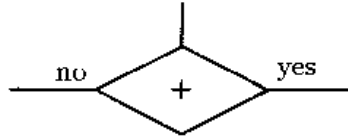
放在 `\savebox` 中的图形元素将会根据盒子被构造时候的单位长度确定尺寸，不会受随后 `\unitlength` 改变的影响。

```

\newsavebox{\testcon}
\savebox{\testcon}(0,0){%
  \thicklines
  \put(0,0.5){\line(-2,-1){1.0}}
  \put(0,0.5){\line(2,-1){1.0}}
  \put(0,-0.5){\line(-2,1){1.0}}
  \put(0,-0.5){\line(2,1){1.0}}
  \put(0,0.5){\makebox(0,0)[b]
    {\put(0,0){\line(0,1){0.5}}}}
  \put(1.0,0){\line(1,0){1.0}}
  \put(-1.0,0){\line(-1,0){1.0}}
  \put(-1.1,0.1){\makebox(0,0)[br]{no}}
  \put(1.1,0.1){\makebox(0,0)[bl]{yes}}}

```

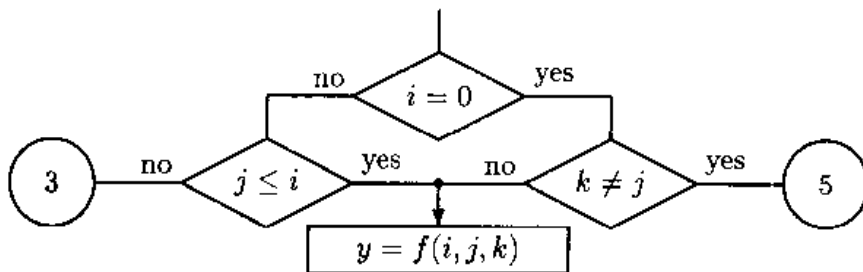
这定义了条件分枝符号, 它经常用在计算机程序的流程图中。



+ 表示符号 `\textcon` 的参考点。

在上面的 `\savebox` 命令中, 参数值 (0,0) 定义了一个零宽度和零高度的盒子, 这样它的中心就放在 `\put` 命令中安置坐标定义的位置。因为我们想要菱形的中心就是这个参考点, 因此菱形上面的竖直线必须没有高度, 否则它的出现就会使得整幅图的中心偏离菱形的中心。因此竖线 (`\line(0,1){0.5}`) 就放在一个零尺寸的 `\makebox` 盒子中。这样整幅图的参考点 (如外面的 `\put` 命令中一致) 就是 + 号所表示的位置。

现在 `\testcon` 命令可以很容易地与其他图形要素组合起来使用。例如,



```

\begin{picture}(10,3) \thicklines
\put(5,2){\usebox{\testcon}\makebox(0,0){$i=0$}}
\put(3,1){\usebox{\testcon}\makebox(0,0){$j\le i$}}
\put(7,1){\usebox{\testcon}\makebox(0,0){$k\neq j$}}
\put(0.5,1){\circle{1.0}\makebox(0,0){3}}
\put(9.5,1){\circle{1.0}\makebox(0,0){5}}
\put(5,1){\vector(0,-1){0.5}\circle*{0.1}}
\put(3.5,0){\framebox(3,0.5){$y = f(i,j,k)$}}
\end{picture}

```

这个例子说明, 在一条 `\put` 命令中可以包含不只一条图形对象。这里把表示符号的 `\usebox{\testcon}` 命令同包含文本的 `\makebox(0,0)` 放在一起。类似地, `\circle{1.0}` 命令分别同其中的居中数字 3 和 5 放在一条 `\put` 命令中。最后, `\vector` 和 `\circle` 命令也放在了一起。当多条命令放在同一 `\put` 命令中时, 在图形元素之间必须不能有空

格。

也可以用 `\savebox` 命令存贮一个完整的 `picture` 环境。这时参数值 (x 尺寸, y 尺寸) 和位置是被忽略的, 因为在 `picture` 环境中就有这些尺寸定义。语法为

```
\savebox{\图形名称}{\begin{picture}(x尺寸,y尺寸)
... ..
\end{picture}}
```

如果这样存贮的图形只做为子图来使用时, (x 尺寸, y 尺寸) 应预设为 (0,0), 这样就可以相对于子图来定位。被保存的图形将会根据外面的 `picture` 环境进行放缩。前面的示例符号 `\testcon` 可以这样保存起来:

```
\savebox{\testcon}{\begin{picture}(0,0) ...
\end{picture}}
```

这里的 ... 代表前一页中定义 `\testcon` 命令时所用的命令集合, 从 `\thicklines` 开始, 到最后一个 `}` 结束。

把子图作为 `picture` 环境保存起来的另一个好处是: 即使点 (0,0) 并不是符号的真正中心, 但也可以把它设成外部 `\put` 命令的参考位置点。这也就是说在 `\testcon` 的定义中, 我们可以把

```
\put(0,0.5){\makebox(0,0)[b]{\put(0,0){\line(0,1){0.5}}}}
```

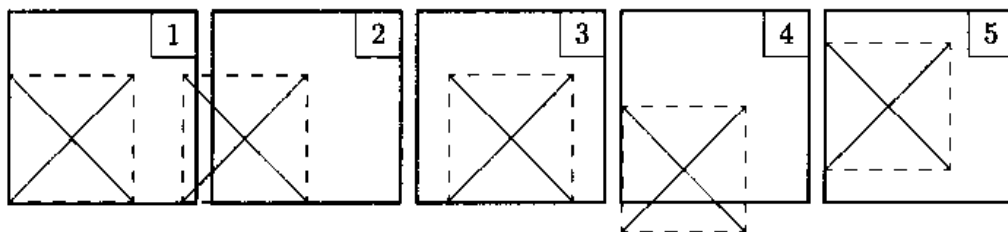
改成 `\put(0,0.5){\line(0,1){0.5}}`。也就是现在不需要“隐藏”会把符号中心从点 (0,0) 移开的竖直线。

6.5.4 图形环境的广义语法

在 `picture` 环境的广义语法中多了一对坐标, 其为可省参数:

```
\begin{picture}(x尺寸,y尺寸)(x偏移,y偏移)
图形命令
\end{picture}
```

在这种形式中, (x 偏移, y 偏移) 定义了左下角的坐标。这就是说对环境中的所有 `\put` 命令, 要从其定位坐标中减去 x 偏移和 y 偏移, 这样整幅图就向左平移了 x 偏移单位, 向下平移了 y 偏移单位。



这里我们给出了五张图形, 每个图形中有一个细线方框, 宽 $3UL$, 高 $3UL$ ($UL=7.2mm$). 在每个方框中, 用 `\put` 加入了一个子图, 其中有 $2 \times 2UL$ 的虚线框及两条对角线。每幅图的构成中除了偏移参数外, 所用命令都是相同的。

- | | | |
|--|----|-----------------------------|
| 1. <code>\put(0,0){\begin{picture}(2,2)(0,0)</code> | 子图 | <code>\end{picture}}</code> |
| 2. <code>\put(0,0){\begin{picture}(2,2)(0.5,0)</code> | 子图 | <code>\end{picture}}</code> |
| 3. <code>\put(0,0){\begin{picture}(2,2)(-0.5,0)</code> | 子图 | <code>\end{picture}}</code> |

- | | | |
|--|----|-----------------------------|
| 4. <code>\put(0,0){\begin{picture}(2,2)(0,0.5)</code> | 子图 | <code>\end{picture}}</code> |
| 5. <code>\put(0,0){\begin{picture}(2,2)(0,-0.5)</code> | 子图 | <code>\end{picture}}</code> |

在第 2 种和第 4 种情形中, 部分子图位于主图边界外面, 当有偏移时必须考虑这一点。

6.5.5 示例

到现在为止, 我们已经通过举例, 相当详细地说明了各种图形元素。然而, 我们有点儿冷落了 `\multiput` 命令, 因此这里给出一些例子说明它的用法。(在 6.3 节中描述了 `\multiput` 命令。)

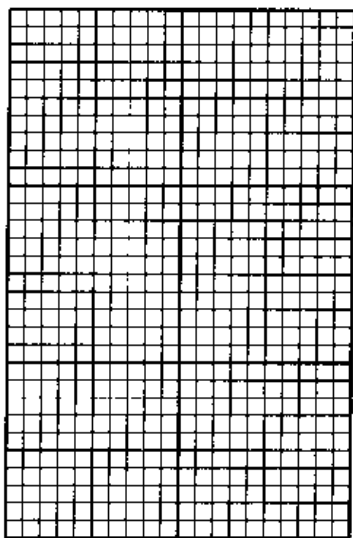
```
\multiput(0,0)(1,2){7}{\circle*{1}}
\multiput(8,0)(2,0){10}{\begin{picture}(0,0)
  \multiput(0,0)(1,2){7}{\circle*{1}}
\end{picture}}
```



第一条 `\multiput` 命令生成七个直径为 1mm 的点, 它们从 (0,0) 开始, 然后每次向右平移 1mm, 同时向上平移 2mm. 第二条 `\multiput` 命令生成 10 个子图, 每个子图相对于前者向右平移 2mm, 而子图本身就是前面第一条 `\multiput` 命令生成的七点。

而在下述网格示例中, 首先画出三条粗 0.25mm, 长 30UL 的竖线, 同样粗细的四条长 20UL 的横线, 它们都是从 (0,0) 开始, 彼此间距 10UL. 面另两条竖线和三条横线粗 0.15mm, 是从 (5,0) 和 (0,5) 开始画。最后我们画出了粗 0.075mm 的 19 条竖线和 29 条横线, 它们彼此间隔 1UL。

例: 网格



```
\setlength{\unitlength}{0.1in}
\begin{picture}(20,30)
\linethickness{0.25mm}
  \multiput(0,0)(10,0){3}{\line(0,1){30}}
  \multiput(0,0)(0,10){4}{\line(1,0){20}}
\linethickness{0.15mm}
  \multiput(5,0)(10,0){2}{\line(0,1){30}}
  \multiput(0,5)(0,10){3}{\line(1,0){20}}
\linethickness{0.075mm}
  \multiput(1,0)(1,0){19}{\line(0,1){30}}
  \multiput(0,1)(0,1){29}{\line(1,0){20}}
\end{picture}
```

6.5.6 扩展宏包

为了增强 `picture` 环境的功能, 标准 $\text{\LaTeX} 2_{\epsilon}$ 版本提供了两个宏包。我们在下面简单讲解一下它们的功能。

去掉某些限制

利用打印机驱动程序的一些功能，可以去掉对斜线长度和倾角的限制。宏包 `pict2e` 就是为了激活特定驱动程序的这种能力而设计的。结果可能不会是真的与设备无关，但应该对于支持这种绘图操作的驱动程序都行得通。

利用这个宏包，圆也可以具有任意直径，而不只是局限于那些可用的特殊圆字体。像通常那样，把这个宏包用 `\usepackage{pict2e}` 包含在导言中。

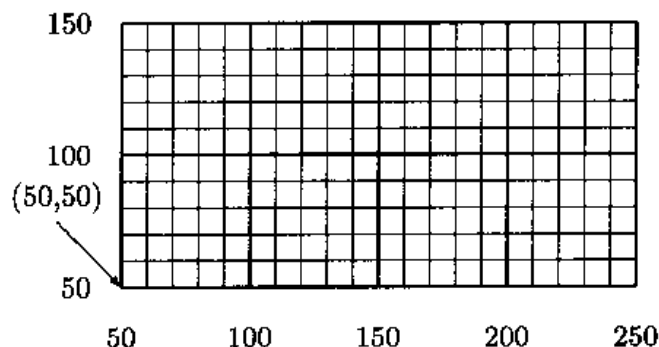
注意：在编写本书时（2001 年 8 月 13 日）这个宏包还没有付诸于实用，因此这里无法给出其选项或驱动程序选择。

格纸

宏包 `graphpap` 增加了一条绘制格纸的命令

`\graphpaper[数](x,y)(lx,ly)`

利用这条命令可以得到下图所示的格纸：



得到上述结果的输入为 `\graphpaper(50,50)(200,100)(UL=0.3mm)`。 `\graphpaper` 命令把方格的左下角放在 (x,y) 处，它有 lx 单位宽， ly 单位高。每隔给定的数单位就划一条格线，而每个第 5 条格纸要粗一些，并且加上标记。如果没有给出数，那就假定它是 10。所有参数值都必须是整数，不能是浮点数。

6.5.7 一般性建议

在实际应用中，每个用户都会形成自己使用 `picture` 环境的风格。这里我们给出的是 [10] 一书中列出的建议，这些建议主要陈述的是 Leslie Lamport 的一些看法。

1. 对于绘制和定位单个图形元素，各种网格尺寸的格纸是相当有用的。利用 `\graphpaper` 命令，用户可以生成任意尺寸的格纸，然后复制足够份，以备将来使用。
2. 网格的最小间距应取为单位长度的大小。这样可以避免定位时出现浮点数。
3. 如果图形中没有斜线，那么图形元素的位置坐标可以直接从格纸上读出来。单个对象的参考点应是一个格点，最好不要位于格点之间。
4. 对于斜线，只有有限的几个斜率（6.4.3 节）。当指定了一个可以接受的斜率后，那么直线的端点应该放在格点上。对箭头也要这样做（6.4.4 节）。（如果包含了 `pict2e` 宏包就不必这样做了。）

5. 图形应该利用子图构成, 或者是子图的子图构成 (6.5.2 节)。这样就可以通过相对于子图定位, 从而简化图形绘制中的这种操作。
6. 经常用的子图应该保存在 `\savebox` 中, 并收集到一个或多个独立的文件中。利用这种方法, 经过几年的积累, 我们可以建立自己的符号库。建立这些符号的文档时应该说明它们的名称、参考点和联系点, 这样其他用户也可以使用它。

在 `picture` 环境一个很小的错误, 也有可能导致毁灭性的后果。因此如果得到的结果完全出乎意料, 也没有必要着急。例如, 若一幅图形设计时是按照单位长度为 `1cm` 进行的, 而没有设定 `\unitlength`, 这样 `LATEX` 就会取默认值 `1pt`, 从而使得图形几乎缩成一点, 其中包含的文本不可想像地都显示在一点上。还有一种常见的情形就是, 图形显示到主图区域的外面, 对于这种出乎意料的结果往往是由于不正确的定位造成的, 可能是错误的符号或者漏掉了小数点。

有时候在水平定位中, 尤其是几个图形用一条 `\put` 或 `\multiput` 命令安置时, 会出现一些小错。这可能是在图形命令之间加了空格。这样的空格也会被当做图形元素, 从而把下一个符号向右进行了一点移位。因此在元素命令之间应该没有空格或回车 (新行)。如果还是有错误, 那就试着换一下安置命令中的元素顺序。如果这样做还行不通, 那就只好每个元素用一条 `\put` 或 `\multiput` 命令了。

6.6 表 格

6.6.1 构造表格

环境 `tabular`, `tabular*` 和 `array` 是生成表格和矩阵的基本工具。这些环境的语法如下:

<code>\begin{array}[位置]{列}</code>	行	<code>\end{array}</code>
<code>\begin{tabular}[位置]{列}</code>	行	<code>\end{tabular}</code>
<code>\begin{tabular*}{宽度}[位置]{列}</code>	行	<code>\end{tabular*}</code>

`array` 环境只能用在 数学模式 (见第五章) 中。之所以在这里提及它, 就是因为它的语法和参数意义与 `tabular` 环境中的完全一样。这三种环境都创建一个小页。参数意义如下:

位置 竖直定位参数 (也可以看 4.7.3 节子段盒子中同名参数的意义)。它可以取下列值:

- t 表格顶部与当前外部文本行的基线对齐;
- b 表格底部与外部基线对齐;

当没有定位参数时, 表格相对于外部基线竖直居中摆放。

宽度 该参数只能出现在 `tabular*` 环境中, 它确定表格的整体宽度。在这种情形中, 列参数必须在第一项后面的某个地方包含 `@`- 表达式 `@{\extracolsep{\fill}}` (细节见下)。对于其他两种环境, 整体宽度是由其文本内容确定的。

列 列格式参数。在这个参数中, 对每列都必须有一个相应的格式符号, 另外还可能包含相应于表格左右边界和列间距等的其他项。

列格式符号可以取下列值:

- l 列内容是左对齐的;

`r` 列内容是右对齐的;

`c` 列内容是居中的;

`p{宽}` 把该列的文本设置成具有给定 宽 的行, 顶行与其他列对齐。实际上文本是用命令 `\parbox[t]{宽}{列文本}` 放在一个子段盒子中的;

`*{数}{列}` 包含在 列 中的列格式被复制 数 份, 因此 `*{5}{|c|}` 与 `|c|c|c|c|c|` 的结果是一样的。

相应于左右边界和列间距的可用 格式化符号 有:

| 画一条竖直线;

|| 画两条紧相邻的竖线;

@{文本} 这一条也叫做 @- 表达式, 它在自己出现的两列之间的每一行上插入文本。

@- 表达式去掉了原本自动加在两列之间的空白。如果在插入文本和后面的列之间需要有空白, 那么就要在 @- 表达式的 文本 中包含 `\hspace{ }` 命令。如果想使某两个特定列之间的距离与其他的标准间距有所不同, 那么可以通过在格式参数中对应于这两列的地方放上 `@{\hspace{宽度}}`, 这样就可能很容易地达到目标了。这里用给定 宽度 的空白取代标准列间距。

在 @- 表达式中的 `\extracolsep{宽}` 会使得后面所有的列间距都增加给定 宽度 的额外间距, 其作用持续到下一个 `\extracolsep` 命令时为止。与标准间距不同, 后面的 @- 表达式并不会去掉这个额外空白。在 `tabular*` 环境中, 在列格式中的某处必须有 `@{\extracolsep\fill}` 命令, 以使得后面所有列间距可以伸展到预定义的表格宽度。

如果表格的左右边界并没有竖线, 那么就会在该处加入等于通常列间距一半的空白。如果不希望加入这个空白, 可以在列格式的开始或结尾处包含一个空的 @- 表达式 `@{ }`, 以删掉这种空白。

行 由表格的实际条目组成, 每一水平行都由 `\\` 结束。这些行由一组彼此之间用 `&` 符号分开的列条目组成。因此每一行应具有与在列定义 列 中相同数目的列条目。有些条目可以是空白的。TeX 把每个列条目当做就好像用大括号 `{ }` 包围起来一样, 因此对于类型样式或尺寸的修改, 将被局限在这个列中。

`\hline` 这条命令只能位于第一行前面, 或者紧接在行结束符 `\\` 后面。它在刚结束的那行下面画一条水平直线。如果这条命令位于表格的开头, 那么就会在表格顶部画一横线, 横线的宽度与表格的宽度相同。

放在一起的两条 `\hline` 命令就会画出两条间隔很小的水平线。

`\cline{n-m}` 该命令从第 n 列的左边开始, 画一条到第 m 列右边结束的水平线。

与 `\hline` 一样, 它也只能位于行结束符 `\\` 的后面, 而且可以同时使用多次。命令 `\cline{1-3}` `\cline{5-7}` 就会在刚结束的行下面画两条水平线, 一条是从第 1 列到第 3 列, 另一条是从第 5 列到第 7 列。在每种情形下, 用的都是完全列宽。

`\multicolumn{数}{列}{文本}` 该命令把接下来的 数 个列组合成单个列, 其宽度等

于总宽度加上列间距。参数列由一个定位参数 l, r 或 c, 以及可能有的 @-表达式和竖线 | 组成。通过把数值的值取 1, 可以改变特定行中某一列的定位参数。

在这种列合列的情况下, 对列参数的分割采用如下方式: 每一列的格式参数都是由定位符号 l, r 或者 c 开始的, 包含所有接下来的内容, 直到遇到另一个定位符号为止。其中第一列还要包含在第一个定位符号之前可能存在的内容。因此 |c@{}r| 就包含三列: 第一列是 |c@{}, 第二列是 r, 第三列是 |。

`\multicolumn` 命令只能位于一行的开始或者一个列分隔符 & 后面。

`\vline` 该命令画一条竖直线, 其高度等于其所在行的行高。用这种命令, 可以得到那些不是贯穿整个表格的竖直线。

对于 `tabular` 和 `array` 环境, 也可以用 $\text{\LaTeX 2}_{\epsilon}$ 命令 `\tabularnewline` 结束一行。这是一条明确的行结束符, 而 `\\` 可以在一个列条目中结束一个文本行。(该命令是在 1994 年 12 月 1 日引进的。)

由于表格是与子段盒子和小页一样的竖直盒子, 因此它也可以与其他的盒子或者文本水平定位 (见 4.7.3 节的例子)。特别要指出的是, 为了使表格在页面上居中, 要利用 `center` 环境:

```
\begin{center} 表格 \end{center}
```

6.6.2 表格样式参数

在表格的生成中, \LaTeX 要利用许多样式参数, 来设置其标准值。用户也可以改变这些值, 既可以在导言中进行全局性改动, 也可以把新值只局限于一个环境中。但不能在表格内部对其进行改动。

`\tabcolsep` 是插入在 `tabular` 和 `tabular*` 环境中两列之间距离的一半;

`\arraycolsep` 是插入在 `array` 环境中两列之间距离的一半;

`\arrayrulewidth` 是表格中水平线与竖直线的粗细;

`\doublerulesep` 是双直线时两线之间的距离。

可以用 `\setlength` 命令像通常那样改变这些参数的值。例如, 为使直线的粗细变为 0.5mm, 可以利用 `\setlength{\arrayrulewidth}{0.5mm}` 命令。除了上面的参数外, 还有参数

`\arraystretch` 可以用来修改表格中的行间距。这是一个放缩因子, 标准值为 1。取值 1.5 就意味着行间距增大 50%。可以用如下命令来重新定义该参数:

```
\renewcommand{\arraystretch}{因子}
```

6.6.3 表格样例

实际上表格的创建要比上面所列的复杂格式简单得多。有几个例子可以很好地演示这一点 ([10])。

最简单的表格就是由行列组成的文本条目, 其每一项要么居中, 要么向一边对齐。

Position	Club	Games	W	T	L	Goals	Points
1	Amesville Rockets	33	19	13	1	66:31	51:15
2	Borden Comets	33	18	9	6	65:37	45:21
3	Clarkson Chargers	33	17	7	9	70:44	41:25
4	Daysdon Bombers	33	14	10	9	66:50	38:28
5	Edbartown Devils	33	16	6	11	63:53	38:28
6	Freeburg Fighters	33	15	7	11	64:47	37:29
7	Gadsby Tigers	33	15	7	11	52:37	37:29
8	Harrisville Hotshots	33	12	11	10	62:58	35:31
9	Idleton Shovers	33	13	9	11	49:51	35:31
10	Jamestown Hornets	33	11	11	11	48:47	33:33
11	Kingston Cowboys	33	13	6	14	54:45	32:34
12	Lonsdale Stompers	33	12	8	13	50:57	32:34
13	Marsdon Heroes	33	9	13	11	50:42	31:35
14	Norburg Flames	33	10	8	15	50:68	28:38
15	Ollison Champions	33	8	9	16	42:49	25:41
16	Petersville Lancers	33	6	8	19	31:77	20:46
17	Quincy Giants	33	7	5	21	40:89	19:47
18	Ralston Regulars	33	3	11	19	37:74	17:49

列宽、列间距以及相应的表格宽度都是自动计算出来的。

上面的表格有八列，第一列是右对齐，第二列是左对齐，第三列居中，接下来的三列又是右对齐，最后两列居中。因此在 `tabular` 环境中的列格式参数是 `{rlcrrrrcc}`，生成这个表格的输入为：

```
\begin{tabular}{rlcrrrrcc}
Position & Club & Games & W & T & L & Goals & Points\\[0.5ex]
1 & Amesville Rockets & 33 & 19 & 13 & 1 & 66:31 & 51:15 \\
2 & Borden Comets & 33 & 18 & 9 & 6 & 65:37 & 45:21 \\
... & ..... & .. & .. & .. & .. & ... & ... \\
17 & Quincy Giants & 33 & 7 & 5 & 21 & 40:89 & 19:47 \\
18 & Ralston Regulars & 33 & 3 & 11 & 19 & 37:74 & 17:49
\end{tabular}
```

在每一行中，各个列条目之间是用符号 `&` 分开的，行本身是用 `\\` 结束的。在第一行后面的 `[0.5ex]` 是增加前两行之间的竖直间距。最后一行不必用结束符号，因为当遇到 `\end{tabular}` 命令时会自动结束该行的。

在列格式参数中可以包含符号 `|`，以使得列之间用竖线分开。把第一行改为

```
\begin{tabular}{r|l||c|rrrr|c|c}
```

那么结果如表 6.1 所示。

表 6.1 表格样例

Position	Club	Games	W	T	L	Goals	Points
1	Amesville Rockets	33	19	13	1	66:31	51:15
2	Borden Comets	33	18	9	6	65:37	45:21
⋮	⋮						⋮
17	Quincy Giants	33	7	5	21	40:89	19:47
18	Ralston Regulars	33	3	11	19	37:74	17:49

在第一个列格式之前或者最后一个列格式之后的符号 | 都会在表格的外边生成一条竖线。两个 || 生成双竖线。可以用 \hline 命令生成与表格宽度相同的水平线。这一命令只能出现在行结束符 \\ 的后面或者第一行的开始。紧接的两条 \hline 命令会绘制双重水平线。输入下述文本:

```
\begin{tabular}{|r|l||c|rrr|c|c|} \hline
Position & Club & Games & W & T & L & Goals & Points\\
\hline\hline
1 & Amesville Rockets & 33 & 19 & 13 & 1 & 66:31 & 51:15 \\
\hline
. . . . .
18 & Ralston Regulars & 33 & 3 & 11 & 19 & 37:74 & 17:49 \\
\hline
\end{tabular}
```

就会得到下面这样的表格:

Position	Club	Games	W	T	L	Goals	Points
1	Amesville Rockets	33	19	13	1	66:31	51:15
2	Borden Comets	33	18	9	6	65:37	45:21
⋮	⋮						⋮
17	Quincy Giants	33	7	5	21	40:89	19:47
18	Ralston Regulars	33	3	11	19	37:74	17:49

在这种情形中, 由于在表格的最后有 \hline, 因此必须给出行结束符 \\。

在这个例子中, 所有行的第三列元素都是一样的, 即 33。这样的公共条目可以利用列格式参数中的 @{文本} 形式的 @- 表达式自动插入, 它把文本插入到相邻两列中。在我们的例子中, 可以把列格式改成

```
{r|@{ 33 }rrrcc} 或者 {r|l||@{ 33 }|rrr|c|c|}
```

这样文本 '33' 就会连同空白一起出现在每行的第二列和第三列之间。这样得到了行条目

稍有点儿不同的同样表格：例如，第四行现在的输入应为

```
4 & Daysdon Bombers & 14 & 10 & 9 & 66:50 & 38:28 \\
```

列格式现在只是由 7 个列定义组成的：rlrrrcc。原来对应第三列的 c 现在已去掉了，从而现在每行应少一个列分隔符 &。现在的第三列，即取胜的局数，是由第二个 & 开始的，而且其与俱乐部名称之间填充进 @{ 33 } 的内容，它不用额外的 & 符号就能自动插入进来。

后面两列以居中的形式给出了得失球与分数的关系。这里的冒号 ‘:’ 只是由于巧合，才使得上下位置一致，因为在所有行中冒号两边都是一个两位数。如果某一项为 9:101，那么由于该项要居中摆放，冒号就会向左稍微偏一点儿了。

也可以实现不依赖于数字位数，而仍然让 ‘:’ 上下对齐，这就要利用在列格式中 r@{:}l 形式的 @- 表达式。其意义为在每一行的一个右对齐和左对齐列之间放入冒号。那么例子中现在的列格式参数变为：

```
{rl@{ 33 }rrrr@{:}lr@{:}l}
```

或者

```
{|rl||@{ 33 }|rrr|lr@{:}l|lr@{:}l|}
```

而行条目变为：

```
4 & Daysdon Bombers & 14 & 10 & 9 & 66 & 50 & 38 & 28 \\
```

先前的一个 c 列，现在被 r@{:}l 形式的两列所取代。@- 表达式就是在相邻两列间插入文本，而且去掉通常情况下应该存在的列间距。因此 r 列是紧靠 ‘:’ 右对齐的，而 l 列是紧靠它左对齐的。

当一列是由包含小数点的长度不定的数字组成时，也可以用同样的方法达到小数点对齐的目的。

原来表示进球数和得分关系的条目现在由关于 ‘:’ 符号定位的两列组成。对于输入得失球或者得分数字时，这没有任何问题。然而，列标题是 ‘Goals’ 和 ‘Points’，其每个占两列的空间，中间没有冒号。可以用命令 \multicolumn 来解决这一问题，它把特定行中所选定的两列合并，并重新定义列格式。无方框的表格中第一行应该是：

```
Position & Club & W & T & L & \multicolumn{2}{c}{Goals}  
& \multicolumn{2}{c}{Points} \\[0.5ex]
```

这里的 \multicolumn{2}{c}{Goals} 意味着接下来的两列被组合成一个居中的列，其中包含着文本 ‘Goals’。对于有框表格，在 \multicolumn 命令中新的格式参数必须是 {c|}，因为当原来的列被合并时，竖线符号 | 也同时被去掉了。要想知道哪些东西属于给定的列，只要记住规则：一列拥有所有的直至下一个 r, l 或 c 为止的内容。

那么我们现在的输入文本为：

```
\begin{tabular}{|rl||rrr|lr@{:}l|lr@{:}l|c|}\hline  
 \multicolumn{10}{|c|}{\bfseries 1st Regional Soccer League ---  
 Final Results 1994/95} \\ \hline  
 &\itshape Club &\itshape W &\itshape T &\itshape L &  
 \multicolumn{2}{c|}{\itshape Goals}
```

1st Regional Soccer League — Final Results 1994/95							
	Club	W	T	L	Goals	Points	Remarks
1	Amesville Rockets	19	13	1	66:31	51:15	League Champs
2	Borden Comets	18	9	6	65:37	45:21	Trophy Winners
3	Clarkson Chargers	17	7	9	70:44	41:25	Candidates for National League
4	Daysdon Bombers	14	10	9	66:50	38:28	
5	Edbartown Devils	16	6	11	63:53	38:28	
6	Freeburg Fighters	15	7	11	64:47	37:29	Medium Teams
7	Gadsby Tigers	15	7	11	52:37	37:29	
8	Harrisville Hotshots	12	11	10	62:58	35:31	
9	Idleton Shovers	13	9	11	49:51	35:31	
10	Jamestown Hornets	11	11	11	48:47	33:33	
11	Kingston Cowboys	13	6	14	54:45	32:34	
12	Lonsdale Stompers	12	8	13	50:57	32:34	
13	Marsdon Heroes	9	13	11	50:42	31:35	
14	Norburg Flames	10	8	15	50:68	28:38	Disbanding
15	Ollison Champions	8	9	16	42:49	25:41	
16	Petersville Lancers	6	8	19	31:77	20:46	Demoted
17	Quincy Giants	7	5	21	40:89	19:47	
18	Ralston Regulars	3	11	19	37:74	17:49	

```
& \multicolumn{2}{c|}{\itshape Points}
```

```
& \itshape Remarks \\ \hline\hline
```

结果如上所示。

在 3-5, 7-14 和 17 行上的水平线是用命令 `\cline{1-9}` 生成的, 而其他地方的则是用 `\hline` 生成的:

```
11 & Kingston Cowboys      & 13 & 6 & 14 & 54&45 & 32&34 &
Medium Teams \\ \cline{1-9}
```

对最后两行需要特别说明一下。备注 'Demoted' 在竖直方向恰好位于两行的中间。这是用如下输入来得到的:

```
18 & Ralston Regulars      & 3 & 11 & 19 & 37&74 & 17&49
& \raisebox{2.3ex}[0pt]{Demoted}\\ \hline
```

`\raisebox` 命令把文本 'Demoted' 向上提升 2.3ex。这里如果没有可省参数 [0pt]。那么这里对盒子的提升会导致最后一行的总高度也增加了 1.5ex。这就会使得在第 17 行下面的水平线与第 18 行文本之间的竖直距离增大。我们就是用可省参数 高度=[0pt] 来抑制这一额外间距的。(参见 4.7.2 节中关于 `\raisebox` 命令的详细描述。)

有时我们需要增大水平直线与包围文本之间的竖直距离。如果前面那个表格的标题为如下形式时, 就会更好看些:

1st Regional Soccer League — Final Results 1994/95						
Club	W	T	L	Goals	Points	Remarks

这可以通过在标题文本中插入一个不可见的竖直标尺, 即支撑 (4.7.6 节) 来做到这一点:

```
\multicolumn{10}{|c|}{\rule[-3mm]{0mm}{8mm}\bfseries 1st
Regional Soccer League --- Final Results 1994/95} \\ \hline
```

这个插入进来的标尺宽度为 0 mm, 所以它是不可见的, 它向基线下面伸展了 3 mm, 有 8mm 高。因此它向基线上面伸展了 5mm。这条命令非常有效地把两个方向的水平线推离了标题。如果一行中不只一列, 那么只要在一列中包含支撑就可以了, 因为整行的尺寸是由最大列决定的。

上面所有的例子中, 在每个列中的条目都只有一行。而有的表格中某些列包含多行文本。例如,

Model	Description	Price
WT5-1P195SSI4	OCTANE/SSI, R10000 195 MHz/1MB cache, 128MB Memory, 4GB System Disk, 20" Monitor	76,492.00
WT5-2P195SSI4	OCTANE/SSI, Dual R1000 195 MHz/1MB cache, 128MB Memory, 4GB System Disk, 20" Monitor	88,382.00
WT5-1P195MXI4	OCTANE/MXI, R10000 195MHz/1MB cache, 4MB texture memory, 128MB Memory, 4GB System Disk, 20" Monitor	91,792.00
WT5-2P195MXI4	OCTANE/MXI, Dual R10000 195MHz/1MB cache, 4MB texture memory, 128MB Memory, 4GB System Disk, 20" Monitor	103,692.00

上面这个表格由三列组成, 第一列左对齐, 第三列右对齐。中间的那列包含着多行文本, 每行宽度为 8.0cm。这是用列格式符号 `p{ 宽度}` 来做到的。在这个表格中全部的列格式参数是 `{lp{8.0cm}r}`。上述表格的输入文本为:

```
\begin{tabular}{lp{8.0cm}r}
\bfseries Model & Description & \bfseries Price \\[1ex]
WT5-1P195SSI4 & OCTANE/SSI, R10000 195 MHz/1MB cache,
128MB Memory, 4GB System Disk, $20''$ Monitor
& 76,492.00\\
. . . . .
$20''$ Monitor & 103,692.00
\end{tabular}
```

中间那列的文本只需简单输入就可以了, 会自动被断成 8.0cm 宽的行。该列就像通常那样用 `&` 符号与其他列分开。

警告：在 p 列中不能用行结束符 \\，因为它会被解释为表格行的结束。而可以用断行命令 \newline 和 \linebreak。然而，如果某一行确实需要用 \\ 来断开，那么整个列就要放在 \parbox 中，其宽度与 p 列的一样。列条目可以由几段组成，空行表示分段。

最后这个例子描述了一个有框表格的空白表。这儿的困难在于设置空白盒子的高度和宽度，因为通常这些量是由文本条目自动确定的。这个例子说明了如何借助于支撑和 \hspace 命令来做到这一点。


Budget Plan 1995–1997							
Project	Nr. <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>			Name <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>			
Year	1995		1996		1997		
	(GB £)	US \$	(GB £)	US \$	(GB £)	US \$	
Investment Costs							
Operating Costs							
Industrial Contract							
Signature				Authorization			

为了得到上述表格，输入文本为

```
\newsavebox{\kk}\newsavebox{\kkk}
\sbox{\kk}{\framebox[4mm]{\rule{0mm}{3mm}}}
\sbox{\kkk}{\usebox{\kk}\usebox{\kk}\usebox{\kk}}
\begin{tabular}{|l|c|c|c|}\hline
\multicolumn{4}{|c|}{\rule[-0.3cm]{0mm}{0.8cm}\bfseries
  Budget Plan 1995--1997}}\hline
\hline
\hline
\rule[-0.4cm]{0mm}{1cm}Project
& \multicolumn{3}{|c|}{Nr. \usebox{\kkk}\hspace{0.5cm}}
\hline
\hspace{0.5cm}Name\usebox{\kkk}\usebox{\kkk}\usebox{\kkk}
\usebox{\kkk}}\hline
\multicolumn{1}{|r|}{Year} & 1995 & 1996 & 1997 \hline
\cline{2-4}
& (GB \pounds) \vline\ US \$ & (GB \pounds) \vline\ US \$
& (GB \pounds) \vline\ US \$ \hline
```



```
Investment & \hspace{2.5cm} & \hspace{2.5cm} & \hspace{2.5cm} \\
Costs      & & & \\ \hline
Operating & & & \\
Costs      & & & \\ \hline
Industrial& & & \\
Contract & & & \\ \hline
\multicolumn{4}{|l|}{\rule[-1.2cm]{0mm}{1.5cm}Signature
    \hspace{5cm}\vline~Authorization} \\ \hline
\end{tabular}
```

输入文本中的前三行只是间接与表格的结构有关。当使用了 `\usebox{\kkk}` 命令时就会画出三个空白框  (见 4.7.1 节)。

我们用 `\hspace{2.5cm}` 命令来设置后三列的宽度, 用 `\vline` 在一列中画一条竖直线, 除这两点外, 这个例子同前面的例子相比, 再没有什么新的地方。这里只需要对最后一行加以解释:

命令 `\multicolumn{4}{|l|}` 把全部四列合并为一列, 文本被设置成左对齐。其文本中首先是一个支撑 `\rule[-12mm]{0mm}{15mm}`, 这也就是说最后这行的高度开始于基线下面 12mm, 总高度为 15mm。接着, 从左页边开始, 显示单词 `Signature`, 间隔 5cm 后是 `\vline`, 即一条竖线。单词 `Authorization` 与竖线之间用一个空白 (~) 隔开。

上面的例子清楚地说明了表格中列宽度和行高度是如何自动确定的。然而这些尺寸也会受支撑和 `\hspace` 命令的影响。而且在 6.6.2 节中描述的命令除了可以改变线粗细外, 还可以增加列间距和行间距。例如,

```
\setlength{\tabcolsep}{5mm}
```

就会在每列的前后插入 5mm 的间距, 即得到了 10mm 的列间距。6.6.2 节中有关于如何使用这些表格样式参数的说明。

6.7 浮动表格和插图

6.7.1 表格的浮动

利用 `tabular` 环境可以在其所出现的地方生成一个表格, 紧接其前面的文本, 后面就是跟着它的其他内容。当在页面上表格与周围文本匹配得很好时, 这没有什么问题, 而且这通常也就是我们想要的。然而, 如果表格很长, 从它定义的地方开始, 在一页中放不下时, 那么就会结束该页, 下一页开头就是这个表格, 后面再接其他文本。这样就会导致当前页面的格式并不怎么令人满意 (空白太大)。

因此我们期望能有这样一种策略, 如果在当前页上表格出现的地方有足够空间放下这个表格, 那么就把它安置在这个地方, 否则就继续排版文本, 保留表格到有足够空间时 (如下一页的开头), 再排版表格。由于表格通常有标题和 / 或题目, 这些项也自然应该随着它一起移动。

L^AT_EX 提供了浮动表格的功能, 利用这种功能, 可以非常好地满足我们的期望。用如下环境可以实现此功能:

```
\begin{table} 上部文本 表格 下方文本 \end{table}
```

这里的 表格 代表由 tabular 等环境定义的完整表格, 上部文本 表示出现在 表格 上方的文本, 而 下方文本 表示出现在 表格 下方的文本。与表格有关联的文本的宽度、间距和位置都要由用户来指定。

出现在 \begin{table} 和 \end{table} 中的所有内容与包围它的外部文本没有多大关系, 其通常都放在当前页的开头。如果已有一个表格占用了这个位置, 那就会尝试是否可能有足够的空间, 把它放在当前页的底部。否则, 就会把它放在下一页上, 这样就有可能积累下来很多没有安放好的表格。包围表格的文本就如同没有表格那样进行排版。

有许多格式参数可以与 table 环境结合使用, 我们将把它们与插图相应的参数一起在 6.7.3 节中介绍。

6.7.2 插图的浮动

picture 环境生成的图形显示在对应于输入文本的地方, 前后都是其他文本。这样就会产生与 tabular 环境同样的问题, 也就是说, 如果当前页上有足够的地方, 那就会得到我们所期望的结果。然而如果插图很高, 在当前页上放不下来, 那么 L^AT_EX 就会结束当前页, 把插图放在下一页的顶部。这样当前页的格式就会很难看。

因此我们还是希望有这样一种机制: 若当前页空间足够, 就把插图放在当前页上, 否则就留到下一页上考虑, 直到找到适当的地方放置。由于插图通常带有标题或说明, 这些东西也应随同移动。

在 6.7.1 节中已引进了相应于表格的这种机制。对插图也有这种机制, 那就是插图的浮动:

```
\begin{figure} 上部文本 插图 下方文本 \end{figure}
```

利用这个环境, 我们就可以与浮动表格一样, 安排插图。关于浮动安置的详情见下节。

6.7.3 浮动安置

L^AT_EX 确实确实地按照上面所要求的那样, 做到了插图和表格连同标题和说明在一起的浮动。这种机制是用下面命令实现的:

```
\begin{figure}[位置] 插图 \end{figure}
\begin{figure*}[位置] 插图 \end{figure*}
\begin{table}[位置] 表格 \end{table}
\begin{table*}[位置] 表格 \end{table*}
```

*- 形式只适用于两列页面格式, 它使得插图或表格占据两列, 而不是正常情形的一列。当页面格式是单列时, 它的作用同标准形式一样。

在上面的语法中, 插图 和 表格 就是要浮动的内容, 它是由 picture 或 tabular 环境, 以及可能存在的 \caption 命令组成。(我们稍后在 6.7.6 节中介绍 \caption 命令。)

而参数值 `位置` 定义了插图或表格允许出现的地方。 `位置` 由零到四个字母组成，因此取值有很多可能，字母的意义如下：

- `h` `here`: 浮动对象可以位于环境输入时所处的地方；它不能用于 `*`-形式；
- `t` `top`: 浮动对象可以出现在当前页的顶部，条件是要有足够的空间以容纳它自己和其前面的文本；如果这行不通，就会把它加到下一页的顶部；其后续文本仍然显示在当前页上，直到正常地分页；（对于两列格式，上面叙述中的页换成列就可以了。）
- `b` `bottom`: 浮动对象可以出现在当前页的底部；对后续文本继续进行排版，直到在当前页上有放下浮动对象的足够地方为止；如果在当前页上已经没有足够地方，浮动对象会被放到下一页的底部；在 `*`-形式中不能取这个值；
- `p` `page`: 浮动对象可以放在一个特殊页（或列）上，该页（或列）上只有插图和表格；
- `!` 与其他字母的组合一起使用，它去掉在 6.7.5 节中描述的关于间距和数值限制。

参数值可以组合，形成几种可能。如果没有给定任何值， \LaTeX 假定它是标准组合 `tbp`。

安置参数值使得定位浮动对象有几种可能，但是实际的插入点是符合下述规则的最早可能点：

- 在它定义的前面一页上再没有浮动对象；
- 插图和表格是按照在文本中定义的先后顺序输出的，因此不可能出现浮动对象在前面定义的同类型对象之前输出的情况；然而插图和表格输出顺序有可能混杂在一起；在两列格式中的双列 `*`-浮动对象也有可能不符合这一顺序；
- 浮动对象只会出现在安置参数值 `位置` 所允许的位置上；若没有该参数值，就会用标准组合 `tbp`；
- 除非在 `位置` 中包含了 `!`，定位要遵从在 6.7.5 节中描述的样式参数的限制；
- 对于组合 `ht`，优先考虑 `h`；即使当前页顶部有足够的空间，浮动对象也会被插入在定义点。

当使用了 `\clearpage`、`\cleardoublepage` 或者 `\end{document}` 命令时，所有还没输出的浮动对象就会显示在单独一页或列上，而不会考虑它的定位参数。

6.7.4 延迟浮动

有的时候可能不希望浮动对象出现在某些页面上，例如不要出现在标题页的顶部。（实际上 \LaTeX 会自动地校正这种情形。）然而，也有时候需要暂时抑制浮动。我们可能希望它位于页面顶部，但是不希望它位于引用它的那节的前面。命令

`\suppressfloats[位置]`

确保在当前页的 `位置` 定义的地方没有其他浮动对象。如果没有可省参数 `位置`，就会抑制所有的浮动；否则 `位置` 可以是 `t` 或 `b`，但不会全有。

注意 `\suppressfloats` 并没有抑制当前页所有的浮动，它只是抑制位于从调用该命令开始到该页结束之间的浮动。因此来自于前一节的浮动仍旧有可能出现在当前页上。

`\suppressfloats` 命令和 `!` 位置参数都是在 \LaTeX 2_ϵ 中才出现的，它们的目的是在尝试提供给作者更大的能力，以控制浮动安置中可能出现的反复无常行为。

6.7.5 浮动中的样式参数

有很多影响浮动安置的样式参数，用户可以修改它们：

`topnumber` 可以位于一页顶部的最多浮动对象数。

`bottomnumber` 可以位于一页底部的最多浮动对象数。

`totalnumber` 不考虑位置，可以位于一页中最多的浮动对象数。

`dbltopnumber` 同 `totalnumber` 一样，只是它限定的是双列格式中横跨两列的浮动对象数。

上述参数都是计数器，可以用命令 `\setcounter{计数器}{数}` 来给它设定新值，这里 计数器 指的是计数器的名称，而 数 是将要设定的新值。

`\topfraction` 是一个浮点数，定义页面顶部多大部分可以放浮动对象。

`\bottomfraction` 是一个浮点数，定义页面底部多大部分可以放浮动对象。

`\textfraction` 该数规定了页面多大部分必须填充以文本。这表示一个最低限度，因此无论顶部，还是底部，总共放浮动对象的部分不能超过 $1 - \text{\textfraction}$ 。

`\floatpagefraction` 在开始新页之前浮动页中填充的浮动对象所占部分的最低限度。

`\dbltopfraction` 同 `\topfraction` 一样，只是它对应于两列页面格式中双列浮动对象。

`\dblfloatpagefraction` 同 `\floatpagefraction` 一样，只是它对应的是两列页面格式中双列浮动对象。

要用 `\renewcommand{命令}{浮点数}` 来改变这些样式参数的值，这里 命令 表示参数名，浮点数 是新的值，每个都必须介于 $[0, 1)$ 之间。

`\floatsep` 出现在页面顶部或底部中的浮动对象之间的竖直距离。

`\textfloatsep` 页面顶部和底部中的浮动对象与文本之间的竖直距离。

`\intextsep` 当用 `h` 安置参数值时，出现在一页中间的浮动对象与上下正文之间的竖直距离。

`\dblfloatsep` 与 `\floatsep` 一样，只是它对应于两列页面格式中双列浮动对象。

`\dbltextfloatsep` 同 `\textfloatsep` 一样，只是它对应于两列页面格式中的双列浮动对象。

这一组样式参数都是弹性长度，可以用 `\setlength` 命令修改它们的值（2.5.2 节）。

`\topfigrule` 在一页顶部浮动对象之后被执行的命令。可以用它加上标尺以把浮动对象与正文分开。但是这里加入的标尺必须是零高度。

`\botfigrule` 类似于 `\topfigrule`，但它是在位于页面底部浮动对象之前被执行。

`\dblfigrule` 类似于 `\topfigrule`，但是是相应于双列浮动对象。

这三条命令通常什么都不做，但必要时可以重定义它们。例如，可以用下面的输入在顶部浮动之后加上粗 0.4pt 的标尺：

```
\renewcommand{\topfigrule}{\vspace*{-3pt}
\rule{\columnwidth}{0.4pt}\vspace{2.6pt} }
```

表 6.2 Computer Center Budget for 1995

Nr.	Item	51505	52201	53998	Total
1.1	Maintenance	130 000		15 000	145 000
1.2	Network costs	5 000		23 000	28 000
1.3	Repairs	25 000	6 000		31 000
1.4	Expendables		68 000		68 000
1.	Total	160 000	74 000	38 000	272 000

由于在 `\vspace*` 中是负的参数值, 整个竖直距离还是零, 这满足命令的要求。

所有单列样式参数在两列页面格式中也有作用, 但它们只适用于填充一列的浮动对象。

6.7.6 浮动对象中的说明

可以用下面的命令生成插图的说明或表格的标题:

`\caption[短标题]{标题文本}`

要把这条命令放在 `figure` 或 `table` 环境中。标题文本 就是与浮动对象显示在一起的文本, 它可以相当长。短标题 是可以省略的, 它是出现在插图或表格列表中的文本 (3.4.4 节)。如果不给出这段文本, 那它就等于 标题文本。如果 标题文本 长度超过 300 个字符, 或者比一行还长的话, 那就应该给出 短标题。

在 `table` 环境中, `\caption` 命令生成形如 ‘Table n : 标题文本’ 的标题, 而在 `figure` 环境中, 则是形如 ‘Figure n : 标题文本’ 的说明, 这里的 n 是自动给出的顺序编号。在文档类 `article` 中, 插图和表格的编号从 1 开始, 直到文档结束。而对于 `report` 和 `book` 类, 每章单独编号, 形式为 $c.n$, 这里 c 表示当前章号, n 是顺序号, 每章开始时, 都重新设为 1。如果用的是 CCT 中文 $\text{\LaTeX} 2_{\epsilon}$ 系统, 说明的前缀进行了必要的汉化, 详见第十章。插图和表格的编号是相互独立的。

如果不想编号的话, 可以不用 `\caption` 命令, 因为包含在浮动环境中的任意文本都会随着其他内容一起移动。而 `\caption` 比普通文本强的地方就在于自动编号, 而且其会自动列在插图和表格列表中。然而, 也可以按 3.4.4 节所讲的那样用下面的命令手工向列表中增加一些项:

`\addcontentsline` 和 `\addtocontents`

当 `\caption` 命令位于浮动环境中其他材料的前面, 那就可以形成一个标题, 即编号和文本显示在表格或插图的上方。如果该命令出现在所有其他浮动命令的后面, 那么就会在对象的下方加上说明。也就是说, `\caption` 只是浮动中的一项, 其中文本出现在顶部 (标题) 或底部 (说明) 与用户放的位置有关。

如果 标题文本 的长度不足一行, 那它就会居中排列, 否则它同正常段落一样。可以通过把命令放在子段盒子或者小页中来相应于表格或插图调整总宽度。例如,

`\parbox{宽度}{\caption{标题文本}}`

下面的例子说明了如何把文本、表格和图形命令组合进浮动对象。

6.7.7 浮动示例

前两个表格是用下述文本生成的：

```
\begin{table} \caption{Computer Center Budget for 1995}
\begin{tabular}{|l|l|l|r|r|r|r|} ... \end{tabular}
\end{table}
```

(这段文本是输入在上一节最后一段的前面。下面的第二个表格就输入在此处。)

```
\begin{table}
\caption{\textbf{Estimates for 1996} {\em A continuation...}}
\begin{tabular}{|l|l|l|r|r|r|r|} .. ... \end{tabular}
\end{table}
```

由于在 `table` 环境中没有给出安置参数值，因此这里用的是标准值 `tbp`。第一个表格被放在该页的顶部，因为它输入的很早（在上一节中），而且那里有足够的空间放下它。然用调用 `\suppressfloats` 命令（6.7.4 节），以禁止在这一页上再出现其他的浮动对象。因此第二个表格就浮动到了现在的位置上。

窄的插图或表格可以并排放在一起，如下例所示。

```
\begin{figure}[ht]
\setlength{\unitlength}{1cm}
\begin{minipage}[t]{5.0cm}
\begin{picture}(5.0,2.5) ... \end{picture}\par
\caption{Left}
\end{minipage}
\hfill
\begin{minipage}[t]{6.0cm}
\begin{picture}(6.0,3.0) ... \end{picture}\par
\caption{Right}
\end{minipage}
\end{figure}
```

这两幅图连同其说明分别放在宽 5cm 和 6cm 的 `minipage` 环境中。两个小页之间用 `\hfill` 间距分开。定位参数值 `t` 使得小页环境的第一行是对齐的（4.7.3 节）。在 `figure` 环境中的所有结构当做一个整体进行浮动。

那么现在就有问题了，既然两幅插图的高度不等，这里要求它们的顶行对齐，那为什么是底边在同一水平线上呢？答案是 `picture` 环境建立起一个 LR 盒子（4.7.1 节），它包含所有的图形命令， \LaTeX 认为这些命令就是一行输入，基线就是图形的底边。在这里的两个小页环境中，`picture` 环境是其中的第一项，因此也就是文本的第一个逻辑行。从而它们的基线就用来进行小页的竖直对齐。

如果要把两个图形沿顶边对齐，那就需要在每个小页环境的 `picture` 环境之前插入

表 6.3 Estimates for 1996 *A continuation of the previous budget is no longer practical since, with the installation of the new computing system in 1995, the operating conditions have been completely overhauled.*

Nr.	Item	51505	52201	53998	Total
1.1	Maintenance	240 000			240 000
1.2	Line costs	12 000	8 000	36 000	56 000
1.3	Training			50 000	50 000
1.4	Expansion	80 000	3 000		83 000
1.5	Expendables		42 000		42 000
1.	Total	332 000	53 000	86 000	471 000

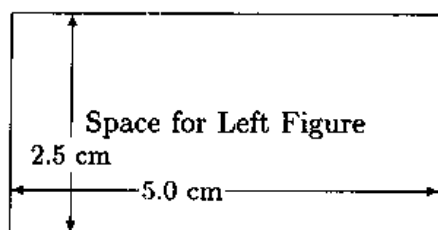


图 6.1 Left

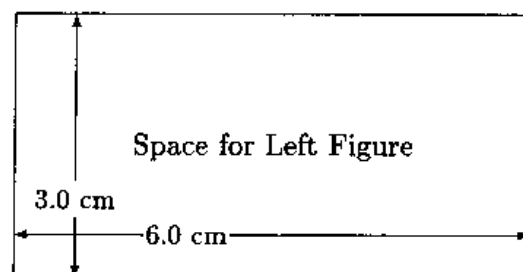


图 6.2 Right

一个没有内容的第一行 (4.7.4 节), 例如可以用 `\mbox{}`。

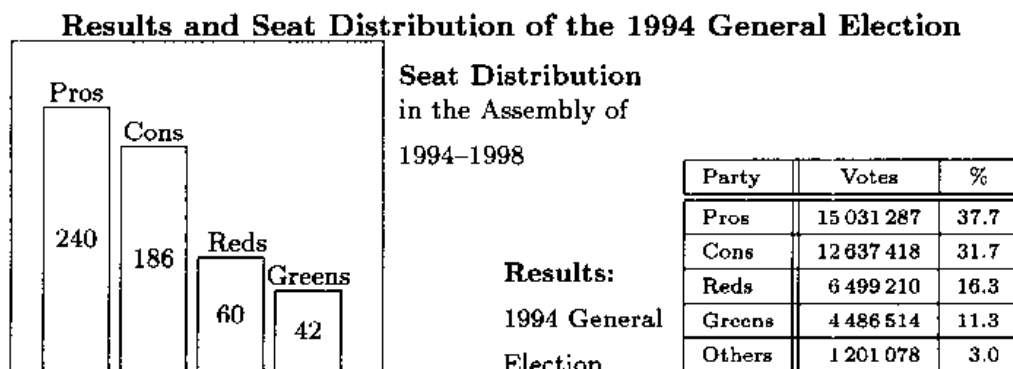
在浮动对象中应用盒子命令是完全可以自由定位的。如果说明文本位于表格或插图的一旁, 而不是上面或下而, 那么对象可以放在小页或子段盒子中, 并选择适当的对齐参数值。下而给出一个例子:

```
\begin{table}[ht]
  \centerline{\bfseries Results and Seat Distribution of the ...}
  \mbox{\small
    \begin{minipage}[b]{7.7cm}
      \begin{minipage}[t]{4.4cm}
        \mbox{}\!\! \unitlength0.75cm
        \begin{picture}(5.75,5.0) ... \end{picture}
      \end{minipage} \hfill
      \parbox[t]{3.2cm}{\makebox[0cm]{}\!\!\bfseries Seat ...} ...
    } \mbox{}
  \end{minipage}
  \hspace{-3cm}
  \begin{minipage}[b]{7cm}
    \parbox[b]{2.5cm}{\bfseries Results:}\!\! General Election...
  \end{minipage}
\end{table}
```

```

\hfill
\begin{tabular}[b]{l|l|r|r|} ... \end{tabular}
\end{minipage} }
\end{table}

```



在这里需要使用嵌套竖直盒子。最左边是由图形和右上角的标题组成，它位于一个宽 7.7cm 的小页中，其中的图形所在的另一个小页宽 4.4cm，而文本又位于一个宽 3.2cm 的子段盒子中。这两者是顶行对齐。为此需要在 `picture` 前面加上一个没有内容的行 `\mbox{}` (见 4.7.4 节)，以使得顶行与子段盒子对齐。

右边是由一个宽 7cm 的小页组成，它向左平移了 3cm，并与左面的小页沿底边对齐。它由一个文本宽 2.5cm 的子段盒子和一个表格组成，表格自动就是一个竖直盒子。这两者是底部对齐的。

6.7.8 在正文中对插图和表格的引用

表格和插图的自动编号，就意味着在写作时作者并不知道它们的编号。面有时候作者又希望能够通过这些对象的编号引用它们，如 ‘Figure 3’ 或 ‘Table 5 illustrates’，因此需要找到一种引用的方法。而仅仅跟踪已调用的 `\caption` 中的编号又是不够的，因为文档可能不是按顺序写作的，修改时又可能插入或删除插图或表格。

这个问题可以借助于 L^AT_EX 的交叉索引系统来解决，在 8.3.1 节中将详细讲解。基本命令是

```
\label{名称} \ref{名称}
```

这里给正文中要用到的插图或表格编号赋予一个关键词 名称。关键词可以是字母、数字或符号的任意组合。赋值是用 `\label` 命令放在 `\caption` 命令的说明文本的任何地方就可以了；在正文中，调用 `\ref` 命令就会插入相应关键词所对应的编号。

下面用一个例子就可以很好地说明这一操作。在第 155 页上的表格标题实际上输入为

```
\caption{\label{budget95} Computer Center ... }
```

因此在正文中用 `Table \ref{budget95}` 可以生成 ‘表 6.2’。

还有另一条引用命令 `\pageref`，它可以生成被引用对象所在那一页的页码。例如，刚刚几行前，就是用 ‘在第 `\pageref{budget95}` 页上’ 得到文本 ‘在第 155 页上’。

第七章 定制 L^AT_EX

在 L^AT_EX 中用户可以定义自己的命令和环境。然而这不可避免地要频繁用到 L^AT_EX 中的记数器和长度，因此我们首先详细讨论一下这些对象，并说明如何使用它们。

7.1 记数器

7.1.1 L^AT_EX 记数器

L^AT_EX 管理着大量的记数器，在启动时给出它们的初始值，通过调用特定命令可以改变它们的值。这些记数器的绝大多数都与可以改变它们的命令有相同的名称：

part	chapter	paragraph	figure	enumi
	section	subparagraph	table	enumii
	subsection	page	footnote	enumiii
	subsubsection	equation	mpfootnote	enumiv

从名称上就可以知道大部分记数器的意义，不需要再解释了。记数器 `enumi` ... `enumiv` 相应于四个层次的 `enumerate` 环境 (4.3.4 节和 4.3.5 节)，而记数器 `mpfootnote` 控制 `minipage` 环境中的脚注编号 (4.10.4 节)。

除这些记数器外，还可能存在用 `\newtheorem` 命令创建的记数器，它也具有与结构类型参数值相同的名称 (4.5 节)。

记数器的值必须是整数，通常是非负的。一条命令可以同时输出几个值，例如当前的 `\subsection` 命令就输出 7.1.1，在这种情况下调用了多个记数器。例如，`\subsection` 命令会给 `subsection` 记数器增 1，并显示 `chapter`、`section` 和 `subsection` 记数器的值，中间用 (半角) 句号分开。同时，命令还会把 `subsubsection` 记数器设为零，即使后者可能没有出现。

7.1.2 用户自定义的记数器

用户可以用下面的命令创建自己的记数器：

`\newcounter{记数器名}[上级记数器]`

这里 `记数器名` 就是刚建立的记数器的名称。它可以是任一字母的组合，只要不与已存在的记数器名称相同就可以了。因此不能用列在上面的 L^AT_EX 记数器或者前面已经定义的记数器名称作为 `记数器名`。可省参数 `上级记数器` 是另一个已经存在的记数器的名称，其作用就在于只要 `上级记数器` 被 `\stepcounter` 或 `\refstepcounter` 命令 (见下面) 增 1，就把新建立的记数器的值重设为零。

当用 `\newcounter` 创建了一个新的记数器后，它的初始值就是零。

`\newcounter` 记数器不能位于用 `\include` 命令 (8.1.2 节) 读进的文件中。因此最好把所有的 `\newcounter` 命令都放在导言中。

7.1.3 改变记数器的值

无论是 L^AT_EX 记数器, 还是用户自定义的记数器, 都可以用下面的命令改变其值:

`\setcounter{记数器}{数}`

这条命令的意义从字面上就可以知道了: 指定的记数器被赋予给定的数值 (应为整数)。

`\addtocounter{记数器}{数}`

利用这条命令, 指定记数器的值增加了给定数值, 数可以是正值, 也可以是负值。

`\stepcounter{记数器}`

指定记数器的值增 1, 同时所有从属记数器 (即所有把这个记数器作为自己上级记数器的记数器) 的值被重设为零 (见上)。

`\refstepcounter{记数器}`

这条命令的效果与 `\stepcounter` 相同, 但它也同时把记数器设为交叉索引命令 `\label` 中的当前记数器 (见 8.3.1 节)。

例如, 最后一条命令可以用在没有 `\caption` 命令的 `figure` 或 `table` 环境中, 这样在正文中也可以引用这些插图或表格的编号。放在浮动环境中的 `\refstepcounter{figure}` 或 `\refstepcounter{table}` 命令也可以使得相应的记数器变为正确的值, 从而可以用 `\label` 命令给它赋予一个关键词 (8.3.1 节)。

记数器的值可以用下面的命令当做一个数值处理:

`\value{记数器}`

这条命令并不改变记数器的值。它通常与 `\setcounter` 或 `\addtocounter` 结合使用。例如, 若用户已经创建了记数器 `mypage`, 那么就可以用命令

`\setcounter{mypage}{\value{page}}`

使它与页码记数器 `page` 取相同的值。

通常 `\protect` 命令可以用来保护脆弱命令, 使它在传送过程中不被破坏, 它同样也可以放在牢固命令前面, 而不会有任何危害。然而 `\value` 命令是一个例外。绝对不要在它前面加上 `\protect` 命令。

7.1.4 显示记数器的值

在记数器中的值可以用下面的命令显示出来:

<code>\arabic{记数器}</code>	以阿拉伯数字显示,
<code>\Roman{记数器}</code>	以大写罗马数字显示,
<code>\roman{记数器}</code>	以小写罗马数字显示,
<code>\alph{记数器}</code>	以小写字母显示,
<code>\Alph{记数器}</code>	以大写字母显示,
<code>\fnsymbol{记数器}</code>	以脚注符号显示。

在命令 `\alph` 和 `\Alph` 中, 数字 1...26 分别对应于字母 a...z 和 A...Z。这就需要用户保证记数器的值位于这个范围内。对于 `\fnsymbol`, 数字 1...9 输出的符号分别是

* ↑ § ¶ || ** ↑↑ ↑↑. 这里也同样要求用户保证计数器的值不要达到或超过 10.

有一些计数器, 还存在下面这样形式的命令:

`\the` 计数器

这里 `\the` 紧接着 计数器 的名称, 如 `\thepage`. 这种命令通常与 `\arabic{计数器}` 是一样的, 但它也可以由几条计数器命令一起组成. 例如, 在文档类 `book` 和 `report` 中, 命令 `\thesection` 就是由章号和节号一起组成的:

`\arabic{chapter}.\arabic{section}`

当前 `\thesection` 的结果就是 7.1.

页码、公式或章节编号等等的自动显示, 都是通过调用适当的 `\the` 计数器 命令完成的. 如果需要另一种不同格式的自动编号, 比如说字母型的公式编号, 相应 `\the` 计数器 命令的定义就可以用 7.3 节中的方法进行修改.

7.2 长 度

我们在前面已经多次指出, 类似于 `\parskip` 或 `\textwidth` 这样的长度参数的值可以用命令 `\setlength` 来设成新值. 有些参数需要取可以伸展和收缩的弹性长度. 这些参数主要用来生成竖直或水平距离. 在 2.5 节中已详细给出了长度单位的类型, 这里不再重复. 本节讨论长度的赋值以及其他的控制长度的命令.

给长度参数赋值的标准 L^AT_EX 方法是用下面的命令:

`\setlength{\长度命令}{长度指定}`

这里 长度指定 既可以是具体的长度 (要有单位), 也可以是另一个长度参数. 在后一种情形中, `\长度命令` 就取这个参数的当前值. 因此在一个 `list` 环境中用

`\setlength{\rightmargin}{\leftmargin}`

就可以使得右页边与左页边的宽度相同.

可以用下面的命令增加长度值:

`\addtolength{\长度命令}{长度指定}`

这条命令就把 长度指定 加到 `\长度命令` 参数上去. 若 长度指定 为负值, 就减去相应的量. 同样, 可以用另一个长度参数作为 长度指定, 参数前面可以有负号, 这样就可以加上或减去这个参数. 在长度参数前面的数值会与参数中的值相乘: `0.5\textwidth` 意味着文本列宽的一半, 而 `2\parskip` 是段间距的两倍.

利用命令

`\settowidth{\长度命令}{文本}`

可以使 `\长度命令` 参数的值等于 文本 处于 LR 模式 (通常是从左到右) 时的自然宽度.

类似地, 命令

`\settoheight{\长度命令}{文本}`

`\settodepth{\长度命令}{文本}`

把 `\长度命令` 的值分别取为文本在基线上方或下方的高度与深度.

最后, 命令

```
\stretch{浮点数}
```

生成一个弹性长度, 其可展性是 `\fill` 的给定浮点数倍 (2.5.2 节)。

用户要自己定义长度, 可以用如下命令:

```
\newlength{\新长度命令}
```

这样就可以建立起长度 `\新长度命令`, 初始值为 0pt. 上面所讲的命令都可以用来处理它的值。

命令

```
\addvspace{长度指定}
```

会在其所处的地方插入等于 长度指定 的额外竖直距离。如果同时多次使用这条命令, 那么实际被插入的间距是其中最大的那个, 而不是所有间距的总和。这条命令只能用在两段之间。把它应用于用户自己定义的命令和环境中, 可以使得生成的结构更像段落。

7.3 用户定义命令

在 L^AT_EX 中可以用下面的命令定义或重定义新的命令:

```
\newcommand {\命令名称}[参数个数][可省参数]{定义}
```

```
\renewcommand{\命令名称}[参数个数][可省参数]{定义}
```

第一条命令是用来定义不存在的新命令。命令名称可以是字母的任意组合, 只要不与已定义的命令重名即可。第二条命令是用来重新定义一条已存在的命令。对这两种情形, 如果调用了不正确的变量, 都会给出一条错误信息。可省参数 参数个数 是一个介于 1 到 9 之间的数, 它规定了新定义的命令或者被改变了定义的命令中有多少个参数值。第二个可省参数值 可省参数 给出了新命令可以为可省参数值取的默认值。命令的实际定义是包含在 定义 中。注意, 在 L^AT_EX 2.09 中, 不存在第二个可省参数。

7.3.1 没有参数值的命令

我们首先演示没有可省参数值 参数个数 的 `\newcommand` 命令的用法。当一种固定的 L^AT_EX 命令或用户命令组合被多次重复时, 就可以用这种形式的命令给它赋一个名称。例如, 结构 x_1, \dots, x_n 称为 x 向量, 经常出现在数学公式中, 它是用数学模式中的 `x_1, \ldots, x_n` 生成的。为此输入

```
\newcommand{\xvec}{x_1, \ldots, x_n}
```

就可以创建一个新的命令, 名称为 `\xvec`。此后就可以同其他命令一样调用这条新定义的命令。当调用它的时候, 它就在自己所处的地方插入相应的文本或命令序列 (即这里的 `x_1, \ldots, x_n`)。事实上, 这里的过程是: 当 `\xvec` 被调用时, 它的定义就进入 L^AT_EX 处理系统中。

由于新的 `\xvec` 命令定义中包含数学命令 (下标命令 `_`), 因此只能在数学模式中调用。从而在文本模式中需要用 `\xvec` 来得到 x_1, \dots, x_n 。从这点来看, 在定义中包含数学模式切换也不失为一个好主意:

```
\newcommand{\xvec}{\$x_1,\ldots,x_n\$}
```

这样 `\xvec` 就生成 x_1, \dots, x_n 。然而, 这样一来, 它就只能用在文本模式中, 而不能用在数学模式中。下面是一种可以保证命令在两种模式中都可以使用的技巧: 把命令定义为

```
\newcommand{\xvec}{\ensuremath{x_1,\ldots,x_n}}
```

这样 `\xvec` 和 `$_xvec$` 都是可以接受的, 而且结果一样。

在 $\text{\LaTeX} 2.09$ 中, 没有 `\ensuremath` 命令。此时在数学模式中可以用 `\mbox{$_\dots$}`, 因为在文本模式中, `\mbox` 是被忽略的, 但在数学模式中, 它就可以暂时切换到文本模式中, 而其中的 `$` 符号又可以激活数学模式。这两种方法定义的命令在有些情况下, 会得到不同的结果。例如, 对于用 `\ensuremath` 命令定义的 `\xvec` 命令, `a_{\xvec}` 的结果就会是 a_{x_1, \dots, x_n} , 而采用 `\mbox` 定义的 `\xvec` 命令, 相应结果为 a_{x_1, \dots, x_n} 。相比之下, 在 $\text{\LaTeX} 2_\epsilon$ 中的 `\ensuremath` 的功能就要强一些。

在文本中应用 `\xvec` 时应该写成 `\xvec{}`, 这是因为 \TeX 认为它是一个没有参数值的命令, 当它遇到第一个非字母字符时就终止其名称。如果这里遇到的第一个非字母字符是空格, 那它就结束命令名称, 但并不插入单词间隔 (2.2 节)。因此 `\xvec and ...` 的结果是 $x_1, \dots, x_n \text{ and } \dots$, 其没有单词间隔。只要在命令名称后面插入一个空格命令 `_` 或者空结构 `{}` 就可以解决这个问题, 因此这里可以用 `\xvec_` 或 `\xvec{}`。

当然也可以在 `\xvec` 定义中就包含空格, 即 `{\ensuremath{x_1,\ldots,x_n}}`。现在跟在命令名称后面的空格还是要被去掉的, 但命令本身会插入一个空格。然而, 在实际中我们不推荐这种做法, 因为这个程序设计中的空格就总是存在的, 即使接在命令后面的是标点符号或其他符号也不例外。

上面各个不同的例子中用的都是命令 `\newcommand`, 但实际上这条命令只能用一次, 以初始化用户定义的还不存在的命令。一旦 `\xvec` 已经被定义好, 那么修正版本只能用 `\renewcommand` 命令来给出。实际上这里的第二个和第三个 `\xvec` 命令就是这样做的。

照这个例子的样子, 用户可以通过 `\newcommand` (或者 `\renewcommand`) 命令来给命令和文本组合起一个新名称, 然后在需要的时候调用它。利用这种方法, 输入量会显著减少, 而且也会减少出错的机会, 当处理的是复杂数学结构的时候, 这一点表现得尤为突出。

如果不确定给命令选择的名称是否已经存在, 那么可以利用

```
\providecommand{\命令名称}[参数个数][可省参数]{定义}
```

进行确认。这条命令的语法与 `\newcommand` 和 `\renewcommand` 命令完全一样。差别就在于如果命令已经存在, 新定义就被忽略, 否则就定义一条新的命令。因此如果在不确定命令是否存在的情况下, 想覆盖命令的当前定义, 可以用如下方法实现: 首先调用 `\providecommand` 确保命令存在, 然后用 `\renewcommand` 命令给出真正的定义。然而, 做这一操作时需要特别仔细!

7.3.2 有参数值的命令

除了 x_1, \dots, x_n 结构外, 在数学中还有 y_1, \dots, y_n 和 z_1, \dots, z_n 等类似的向量结构。因此我们可以按 `\xvec` 的样子定义命令 `\yvec` 和 `\zvec`。然而, 我们也可以定义一个更

一般的向量命令，把变化部分作为参数值。对于当前这个例子，变化部分是字母 x, y 和 z 。只有一个变量的命令是用可省参数值 [1] 生成的。因此，

```
\newcommand{\avec}[1]{\ensuremath{\#1_1,\ldots,\#1_n}}
```

就定义了一般性的向量命令 `\avec{参数值}`。这样当调用 `\avec{x}` 时结果就是 x_1, \dots, x_n ，而 `\avec{y}` 显示 y_1, \dots, y_n 。在命令定义中的字符 #1 只是一个哑元，表示当命令被调用时，要用 参数值 的文本取代所有出现的 #1。只要把定义中所出现的 #1 都想像成 x 或 y ，就相当容易地理解所定义的结构了。

在哑元 #1 中的数字 1 在这里好像没有什么用处。事实上，对于只有一个参数值的命令，它确实没有什么意义。然而，对于多参数值的命令，它的作用就很明显了。比如说，我们要定义一条命令，它既能生成结构 x_1, \dots, x_n ，也能生成 v_1, \dots, v_m 。这就需要有二个参数值，一个用来指定 u, v 等，另一个用来确定最后的那一个下标为 n, m 等。这样的命令是如下生成的：

```
\newcommand{\anvec}[2]{\ensuremath{\#1_1,\ldots,\#1_{\#2}}}
```

这样 `\anvec{u}{n}` 就得到 u_1, \dots, u_n ，而 `\anvec{v}{m}` 得到 v_1, \dots, v_m 。`\newcommand` 命令中的可省参数值 [2] 说明要定义的命令中包含二个参数值；在定义部分，#1 表示第一个参数值，#2 表示第二个参数值。把 #1 想像成 u 或 v ，把 #2 想像成 n 或 m ，那么就可以很容易知道 `\anvec{参数 1}{参数 2}` 的操作方式了。

这种方式很容易推广到有更多参数值的命令定义。做了如下定义后，

```
\newcommand{\subvec}[3]{\ensuremath{\#1_{\#2},\ldots,\#1_{\#3}}}
```

命令 `\subvec` 就具有三个参数值。从其定义易知 `\subvec{a}{i}{j}` 生成 a_i, \dots, a_j 。

只由单个字符构成的参数值不必放在大括号 { } 内，可直接给出。如果它是第一个参数值，那必须像通常那样，用空格把它与命令名分开。因此 `\subvec aik` 与 `\subvec{a}{i}{k}` 的结果是相同的，而 `subvec xin` 生成与前面例子 `\xvec` 同样的结构 x_1, \dots, x_n 。

当参数值不只由一个字符组成时，它就必须放在大括号 { } 内，因为这里的大括号表示要把参数值内容当做一个整体处理。因此 `\subvec{A}{ij}{lk}` 的结果为 A_{ij}, \dots, A_{lk} 。这里的参数取代是 A 相应于 #1， ij 相应于 #2， lk 相应于 #3。

可是为什么 `\subvec{A}{ij}{lk}` 的结果是 A_{ij}, \dots, A_{lk} ，而不是所期望的 A_{ij}, \dots, A_{lk} 呢？这是因为虽然在大括号内的参数值被当做一个整体替换进定义文本中，但大括号自身并没有被加进。这里替换后的命令文本实际上类似于 `\ensuremath{A_{ij},\ldots,A_{lk}}`，因此实际上只有接在下标符号 $_$ 后面的第一个字母被降低。为了使两个字母都被降低，那在命令文本中它们就必须做为一个整体出现，也就是类似于 `A_{ij},\ldots,A_{lk}`。在 `\subvec` 命令的参数值中额外再加一对大括号就可以做到这一点，即 `\subvec{A}{\{ij\}}{\{lk\}}`。而更好的解决方法是一开始就在定义中加上大括号：

```
\newcommand{\subvec}[3]{\ensuremath{\#1_{\#2},\ldots,\#1_{\#3}}}
```

这样即使每个参数值只有一层大括号，也能得到所期望的结果：

`\subvec{A}{\{ij\}}{\{lk\}}` 的结果为 A_{ij}, \dots, A_{lk} 。

7.3.3 有一个可省参数值的命令

我们知道很多 \LaTeX 命令可以有可省参数值, 其中最典型的例子就是 \newcommand 命令自身。在 $\text{\LaTeX}_{2\epsilon}$ 中, 同样也可以使用用户定义的命令有一个可省参数值。这样做的好处就在于, 虽然多提供了一个参数, 但在绝大多数情况下它只取标准值, 不需要显式改变其值。

例如, 在上一节用户定义的命令 \subvec 中有三个参数值, 分别相应于字母和第一个及最后一个下标。然而, 通常字母就是 x , 因此把它做为一个可省参数不失为一个好主意, 这样只有在字母不同时才需要指定。要做到这一点, 利用如下命令:

```
\renewcommand{\subvec}[3][x]{\ensuremath{\#1_{\#2},\ldots,\#1_{\#3}}}
```

它与前面定义的区别就在于 $[3]$ 参数值后面多了 $[x]$ 。这就说明了三个参数值中的第一个是可省的, 其标准值为 x 。现在 $\text{\subvec}\{i\}\{j\}$ 的结果就是 x_i, \dots, x_j , 而 $\text{\subvec}[a]\{1\}\{n\}$ 的结果为 a_1, \dots, a_n 。

在用户定义的命令中只能有一个可省参数, 而且也必须是第一个, 即定义中的 $\#1$ 。

7.3.4 用户定义命令的其他样例

在上面用户定义命令的解释中, 我们用向量结构这样很简单的情形做为示例。下面我们演示的是更复杂的情况, 其中要应用到计数器、长度, 甚至一些特殊的 \TeX 命令。

例 1 在 5.4.7 节中, 我们提到 \TeX 命令 \atop 和 \choose 是相当有用的数学命令, 即使在 \LaTeX 中也不例外。不幸的是, 这两条命令的语法同类似的 \LaTeX 命令 \frac 大相径庭。然而,

```
\newcommand{\latop}[2]{\#1\atop \#2}
```

```
\newcommand{\lchoose}[2]{\#1\choose \#2}
```

定义的两条命令 \latop 和 \lchoose 就生成与 \TeX 命令相同的结果, 而且语法也符合通常的 \LaTeX 结构化要求。注意这里为了保证存在必要的大括号, 定义文本中用了两重大括号, 如果只用一重大括号, $\text{\$a\latop\{1\}\{2\}b\$}$ 的结果就是 $\frac{a1}{2b}$ 。

例 2 在下面我们要定义两条命令, $\text{\defbox}\{\text{取样文本}\}$ 会设置一个盒子, 其宽度等于取样文本 的长度。然后调用 $\text{\textbox}\{\text{文本}\}$ 命令, 就会在一个宽度与 取样文本 相同的有框盒子中居中显示 文本。

```
\newlength{\wdth}
```

```
\newcommand{\defbox}[1]{\settowidth{\wdth}\#1}
```

```
\newcommand{\textbox}[1]{\framebox[\wdth]\#1}
```

首先创建了一个新的长度参数 \wdth , 然后定义 \defbox , 使得 \wdth 就等于其参数值的长度 (7.2 节), 最后用 \textbox 生成一个相同宽度的有框盒子, 其中文本居中摆放。

```
as wide as this text\\
```

```
\defbox{as wide as this text}\textbox{}\
```

```
\textbox{text}\
```

```
\textbox{longer text}
```

```
as wide as this text
```

text
longer text

例 3 我们要定义一条新的脚注命令 `\myftnote`, 它与通常的 `\footnote{文本}` 命令一样把文本放到脚注所处的地方, 但这里不是用数字做脚注标记, 而依次采用符号 * † ‡ § ¶ || ** †† ‡‡, 在每一新页上都是从 * 开始。为此, 首先需要创建一个新的计数器, 每当 page 计数器增 1 时, 它都要被重置为零。做法为 (见 7.1.2 节):

```
\newcounter{myfn}[page]
```

这样就可以使用户自定义的计数器 myfn 每当 page 增 1 时都会自动重置为零。下面这条命令

```
\renewcommand{\thefootnote}{\fnsymbol{footnote}}
```

重定义脚注标记为计数器 footnote 所对应的上述符号 (4.10.2 和 7.1.4 节)。现在可以如下构造新的脚注命令:

```
\newcommand{\myftnote}[1]{\setcounter{footnote}{\value{myfn}}%
\footnote{#1}\stepcounter{myfn}}
```

它就会生成所期望的结果。用户定义的命令 `\myftnote` 拥有一个参数值, 在把 L^AT_EX 计数器 footnote 的值设成与用户定义计数器 myfn 有相同的值后, 就把这个参数值传送给 L^AT_EX 的 `\footnote` 命令。一旦执行了 `\footnote` 命令, 就用 `\stepcounter{myfn}` 命令给计数器 myfn 增 1。但是一旦 page 计数器增 1, 即生成一个新页, 这个计数器就要被重置为零。

例 4 我们下面定义命令 `\alpheqn`, 一旦调用了这条命令, 后续公式将具有相同的编号, 只是后缀字母 a, b, ... , 中间用连字符 '-' 分开。命令 `\reseteqn` 把编号框架重设为原来的样式。因此公式编号序列的形式可能为: 4, 5, 6-a, 6-b, 7。

```
\newcounter{saveeqn}%
\newcommand{\alpheqn}{\setcounter{saveeqn}{\value{equation}}%
\stepcounter{saveeqn}\setcounter{equation}{0}%
\renewcommand{\theequation}
{\mbox{\arabic{saveeqn}-\alph{equation}}}}%
\newcommand{\reseteqn}{\setcounter{equation}{\value{saveeqn}}%
\renewcommand{\theequation}{\arabic{equation}}}}%
```

由于我们在 7.1 节中已讲解了上面所用的命令和计数器, 因此这个例子是很容易理解的。计数器 equation 的当前值被保存到计数器 saveeqn 中, 然后增加 saveeqn, 而 equation 被重置为零。公式编号的形式 `\theequation` 利用这两个计数器进行了重定义。这样公式编号的方式就同原来一样, 而 saveeqn 没有改变。重设命令 `\reseteqn` 就把 saveeqn 的值重新赋给 equation, 并恢复 `\theequation` 的定义。

这条命令只适合用在 article 文档类中。如果所用文档类是 book 或者 report, 那么 `\theequation` 的定义是

```
\arabic{chapter}.\arabic{equation}
```

这里需要进行的必要修改就留给读者做为练习。

在定义组合公式编号的第一个 `\renewcommand{\theequation}` 命令中的 `\mbox` 命令

是必须的, 因为结果要用数学模式显示, 这样连字符 ‘-’ 要被解释成二元运算符 (负号), 其前后与两个被操作数 `\arabic{savaeqn}` 和 `\alph{equation}` 之间就会有额外的间距。因此结果可能是 $6 - a$, 而不是 $6-a$ 。`\mbox` 命令就是为了暂时从数学模式切换进 LR 模式。

例 5 最后我们利用某些很基本的 \TeX 命令, 给出一个例子, 这里无法详细解释这些命令。对于那些输入文本中经常包含化学公式的文档, 这里定义的命令就是非常有用的。

在 5.4.10 节中曾指出化学方程式 $\text{Fe}_2^{2+}\text{Cr}_2\text{O}_4$ 中的下标不在同一水平线上, 而且在文本模式中输入短公式的操作也很复杂。下面这条命令就可以解决这些问题。

```
\newlength{\fntxvi} \newlength{\fntxvii}
\newcommand{\chemical}[1]
{(\fontencoding{OMS}\fontfamily{cmss}\selectfont
  \fntxvi\the\fontdimen16\font
  \fntxvii\the\fontdimen17\font
  \fontdimen16\font=3pt\fontdimen17\font=3pt
  $\mathrm{#1}$
  \fontencoding{OMS}\fontfamily{cmys}\selectfont
  \fontdimen16\font=\fntxvi \fontdimen17\font=\fntxvii)}
```

现在 `\chemical{Fe_2^{2+}Cr_2O_4}` 的结果就是正确的形式 $\text{Fe}_2^{2+}\text{Cr}_2\text{O}_4$ 。

解释: \TeX 命令 `\fontdimen n` 描述了字符字体的特定性质, 如 $n = 16$ 和 $n = 17$ 就确定指标 (下标) 的位置。在用 `\selectfont` (8.5.1 节讲到) 结尾的那行上的命令使得当前数学符号字体连同其内部设计尺寸一起存贮在 \TeX 命令 `\font` 中。当前尺寸 (这里是 10pt) 下数学公式中的所有指标都一致降低 3.0pt。由于对 `\fontdimen` 的改变总是全局性的 (当从一个真正的或者隐式的环境中退出时, 并不恢复以前的值), 因此有必要首先保存当前值, 然后再手工恢复。

7.3.5 条件文本

有经验的 \TeX 用户对在 \TeX 和 \LaTeX 中都可以用的条件文本是相当熟悉的。然而, 条件文本的使用方法并不是很简单明了的, 需要对 \TeX 深层机制有相当的了解。Leslie Lamport 提供了一个名为 `ifthen` 的宏包, David Carlisle 把它进行了推广, 从而使得它可以应用于 $\text{\LaTeX} 2_{\epsilon}$ 。该宏包不但简化了条件文本的使用方法, 而且符合 \LaTeX 语法。

这个宏包可以像通常那样在导言中用下面命令调用:

```
\usepackage{ifthen}
```

这样就可以使用 `\ifthenelse` 和 `\whiledo` 这两条命令了, 它们的语法是:

```
\ifthenelse{ 测试条件 }{then 文本 }{else 文本 }
\whiledo{ 测试条件 }{do 文本 }
```

在这两个语法中, 测试条件 是一个逻辑声明 (下面解释); 在第一条命令中, 如果这个声明等于 $\langle true \rangle$, 那么就把 then 文本 插入到正文中, 否则就把 else 文本 插入到正文中。在第二条命令中, 只要 测试条件 等于 $\langle true \rangle$, do 文本 被插入 (执行)。(do 文本 必须改

变对测试条件的输入, 否则它永不会停止!) 文本中也可以包含命令, 甚至可以定义或重定义命令。

共有四种类型的逻辑声明, 可以把它们组合起来, 以形成复杂的声明。

测试数字

要比较两个数字, 或者值为数字的命令, 只要在它们中间放上关系运算符 $<$, $=$ 或 $>$ 就可以了, 这三个符号分别代表小于、等于或大于。计数器中的值可以通过把它的名称作为 `\value` 命令的参数值来进行测试。例如:

```
\newcommand{\three}{3}
\ifthenelse {\three = 3} {O.K.} {What?}
\ifthenelse {\value{page} < 100 }
{Page xx} {Page xxx}
```

在第一种情形中显示出 ‘O.K.’, 因为 `\three` 等于 3; 在第二种情形中, 如果当前页码小于 100, 就会显示出 Page xx, 否则就显示出 Page xxx。(上面输入文本中加的空格是为了明了, 因为参数值中间的空格总被忽略。)

要测试一个数是偶数, 还是奇数, 可以用 `\isodd` 命令:

```
\ifthenelse {\isodd{\value{page}}}
{odd} {even}
```

测试文本

若要测试两条命令得到的是否是相同一块文本, 或者一条命令是否定义成特定的字符串, 可以用

```
\equal{字符串一}{字符串二}
```

这里的字符串一和字符串二就是文本或者可以简化为文本的命令。例如, 给出如下定义后:

```
\ifthenelse {\equal{\name}{Fred}} {Fredrick} {??}
```

若 `\name` 已被定义成 Fred(用命令 `\newcommand`), 那么就会插入 Fredrick, 否则就显示出两个问号。

测试长度

另一个逻辑声明用来比较两个长度。

```
\lengthtest{关系}
```

这里的 关系 由两个长度或长度命令组成, 中间用关系运算符 $<$, $=$ 或 $>$ 分开。例如,

```
\newlength{\horiz} \newlength{\vert}
\newlength{\min}
. . . . .
\ifthenelse {\lengthtest{\horiz > \vert}}
{\setlength{\min}{\vert}} {\setlength{\min}{\horiz}}
```

会把 `\min` 取成 `\horiz` 和 `\vert` 两者中较小的那个。

测试布尔变量

所谓布尔变量就是要么为 `<true>`，要么为 `<false>` 的参数，也称为 标志。有三条命令可以处理标志：

<code>\newboolean{字符串}</code>	创建一个新的布尔值
<code>\setboolean{字符串}{逻辑值}</code>	赋值 true 或 false
<code>\boolean{字符串}</code>	测试其值

其中最后那个就可以用于 `\ifthenelse` 和 `\whiledo` 命令中的 测试条件。

在 \LaTeX 中还有很多内部布尔值，可以对它们的值进行检测（但决不要重设它们的值！）。其中最有用的就是 `@twoside` 和 `@twocolumn`，可以用来测试当前激活的是不是双面模式或者两列模式。由于其中包含字符 `@`，因此只能直接用在类或宏包文件中，若要用在 \LaTeX 文档中，就要把它们包围在 `\makeatletter` 和 `\makeatother` 之间，见第 101 页上的示例。

组合逻辑声明

可以把上面列出的逻辑声明利用下面的逻辑运算符，组合成更复杂的声明：

`\and` `\or` `\not` `\(` `\)`

熟悉布尔逻辑的读者对这些运算符的含义是非常清楚的。例如，如果激活了两列模式或者 `\paperwidth` 大于 15cm 且同时页码计数器的值小于 100 时，就把 `\textwidth` 的值取为 10cm，那么下面的定义：

```
\ifthenelse {\lengthtest{\textwidth > 10cm} \or
  \(\ \lengthtest{\paperwidth > 15cm} \and
    \vaule{page} < 100 \) }
  {\setlength{\textwidth}{10cm}} {}
```

就可以达到这个目的。

上面这个条件比较复杂，就非常适用于定义可以有不同行为的新命令，或者包含在宏包和类文件中。

下面给出一个相对简单的例子：假设作者不知道出版者用的到底是英国拼写，还是美国拼写。那么他可以在文档中把两种拼写都包含进去，在导言中加进一个布尔变量，以得到最后的选择。

```
\newboolean{US}
\setboolean{US}{true} %For American spelling
%\setboolean{US}{false} %For British spelling
\newcommand{\spell}[2]{\ifthenelse{\boolean{US}}{#1}{#2}}
```

那么现在 `\spell` 命令就会显示出根据标志 `US` 的设置来确定显示第一个还是第二个参数值。因此在正文本中作者可以如下输入：

```
... the \spell{color}{colour} of music ...
```

这样如果指定的是 `\setboolean{US}{true}`, 就会生成美国拼法, 否则得到的就是英国拼法。事实上, 不同工作部分之间可以用不同的拼法, 只需简单地在适当地方改变开关的值就可以了。(在写书时这就是一种好方法, 因为编辑在最终决定手稿之前是有可能改变拼法的。)

下面给出一个利用 `\whiledo` 的例子: 作者希望把某一文本重复写 n 遍, 这里的文本和 n 是可变的, 那么可以用如下方法:

```
\newcounter{mycount}
\newcommand{\replicate}[2]{\setcounter{mycount}{#1}
\whiledo{\value{mycount}>0}{#2\addtocounter{mycount}{-1}}}
```

那么 `\replicate{30}{?}` 就会显示出 30 个问号。

7.4 用户定义的环境

可以用下面的命令来创建或修改环境:

```
\newenvironment {环境名} [参数个数] [可省参数]
{开始的定义} {结束的定义}
\renewenvironment{环境名} [参数个数] [可省参数]
{开始的定义} {结束的定义}
```

这里参数值意义如下:

环境名: 对于 `\newenvironment`, 它不能与已存在的 L^AT_EX 预定义或用户自定义的环境或命令重名。另一方面, 对 `\renewenvironment` 而言, 就必须有同名的环境存在。要对 L^AT_EX 环境进行任何修改, 都要求用户对 L^AT_EX 的内部工作机理有相当深入的了解。

参数个数: 介于 1 到 9 之间的数, 说明环境有多少个参数值; 如果忽略了这个可省参数值, 环境就没有参数值。

可省参数: 如果第一个参数值 (#1) 是可省的, 这是它的默认值; 它与 `\(re)newcommand`(162 页) 中的同名参数值行为一样。

开始的定义: 当 `\begin{环境名}` 被调用时, 被插入的初始化文本; 如果这一文本中包含形如 $\#n$, $n = 1, \dots$, 参数个数 项, 那么环境就要用如下形式调用:

```
\begin{环境名}{参数 1}...{参数 n}...
```

在开始的定义中出现的每个 $\#n$ 都要用 参数 n 取代。

结束的定义: 当调用了 `\end{环境名}` 时, 要插入的结束文本; 这里不要用哑参数值 $\#n$, 因为它们只允许出现在开始的定义文本中。

7.4.1 没有参数的环境

同用户自定义命令中的情形一样, 这里首先讲的环境也是没有可省参数值 参数个数的。用户若要定义一个自己的环境 `sitquote`, 可以如下输入:

```
\newenvironment{sitquote}{\begin{quote}\small
```

```
\itshape}{\end{quote}}
```

在这里 开始的定义 由命令序列 `\begin{quote}\small\itshape` 组成, 而 结束的定义 只有 `\end{quote}`。那么现在调用

```
\begin{sitquote} 文本 \end{sitquote} 就等价于
```

```
\begin{quote}\small\itshape 文本 \end{quote}
```

这样就得到了所期望的结果。例如, 下述结果就是用刚定义的环境生成的:

which sets the text appearing between \begin{sitquote} text \end{sitquote} in the typeface \small\itshape, and indented on both sides from the main margins, as demonstrated here.

这个例子不是很好, 因为只要在 `quote` 环境的开始加上 `\small\itshape` 就很容易得到同样的效果。然而, 如果在文档中经常出现这样的结构, 那么这个新环境对于简化输入和减少在输入 `\small\itshape` 时出错的机会方面就非常有效了。

我们现在把前面这个例子做如下的推广:

```
\newcounter{com}
\newenvironment{comment}
{\noindent\slshape Comment:\begin{quote}\small\itshape}
{\stepcounter{com}\hfill(\arabic{com})\end{quote}}
```

这里 开始的定义 就是由文本和命令组成:

```
\noindent\slshape Comment:\begin{quote}\small\itshape
```

而 结束的定义 组成为:

```
\stepcounter{com}\hfill}{\arabic{com})\end{quote}
```

其中 `com` 是一个由 `\newcounter` 创建的用户计数器。由于 `\begin{comment}` 命令要在环境开始处插入 开始的定义 文本, `\end{comment}` 在结束处插入 结束的定义 文本, 因此很容易知道:

```
\begin{comment} This is a comment.
Comments should ...
... in round parentheses.
\end{comment}
```

的结果应是如下结构:

Comment:

This is a comment. Comments should be preceded by the word Comment: the text being in a small, italic typeface, indented on both sides from the main margins. Each comment receives a running comment number at the lower right in round parentheses. (1)

读者应仔细对比一下环境定义中的命令序列与例子中文本位置的关系, 这样可以准确地知道这个环境的效果。这里有两个非常明显的缺陷, 其一就是当调用 `\begin{comment}` 时该环境正好位于一行的中间, 即前面没有空行的情形, 这时文本 `Comment:` 就会位于当前行尾部, 而注释文本在新段落上开始。其二就是注释的最后一行太长, 使得该行没有足够的空间把活动编号放在行尾。

下面这个修正版本就解决了这两个问题:

```
\renewenvironment{comment}
{\begin{sloppypar}\noindent\slshape Comment:
\begin{quote}\small\itshape}
{\stepcounter{com}\hspace*{\fill}{\arabic{com}}}\end{quote}
\end{sloppypar}}
```

这里利用 `\begin{sloppypar}` 命令使得环境总是在新的段落上开始, 而且在断行时不会有满行的情形发生。如果注释的编号在最后一行放不下来的话, 就会开始一个新行, 编号是右对齐的, 因为这里用了命令 `\hspace*{\fill}`。这里同样希望读者仔细查看插入在 `\begin{comment}` 和 `\end{comment}` 之间的定义文本, 以体会其作用。

7.4.2 有参数的环境

向环境中传递参数值, 同命令中的情形是完全一样的。举例来说, 我们对上面的注释环境进行修正, 使得注释人的姓名加在单词 `Comment:` 的后面; 当环境被激活时这个姓名就是其参数值。

```
\renewenvironment{comment}[1]
{\begin{sloppypar}\noindent\slshape Comment: #1
\begin{quote}\small\itshape}
{\stepcounter{com}\hspace*{\fill}{\arabic{com}}}%
\end{quote}\end{sloppypar}}
```

那么现在输入

```
\begin{comment}{Tom William} This is a modified ...
... environment argument \end{comment}
```

结果就是:

Comment: Tom William

This is a modified comment. Comments should be preceded by the word Comment:, followed by the name of the commenter, with the text of the comment being in a small, italic typeface, indented on both sides from the main margins. Each comment receives a running comment number at the lower right in round parentheses. The name of the commenter is transferred as an environment argument. (2)

下面再对这个例子进行修正, 交换注释编号与注释人姓名的位置。把活动编号放在单词 `Comment:` 后面是没有任何问题的, 因为这只需要简单地把来自于 {结束的定义} 命令插入在上面 #1 哑元参数值的地方。然而, 要把 #1 符号放在原来注释号所处的地方, 在 L^AT_EX 处理过程中就会出现错误信息, 因为这与 `\newenvironment` 命令的语法冲突: 在 {结束的定义} 中不能有哑元参数值。如果哑元参数值位于 `\begin{quote}` 的后面, 那么其位置就不是所期望的地方, 而是处于注释文本的开始。

下面给出解决这个问题的技巧:

```
\newsavebox{\comname}
```

```

\renewenvironment{comment}[1]
  {\begin{sloppypar}\noindent\stepcounter{com}\slshape
   Comment \arabic{com}\sbox{\comname}{#1}
   \begin{quote}\small\itshape
   {\hspace*{\fill}\usebox{\comname}\end{quote}\end{sloppypar}}

```

我们在 4.7.1 节中已讲述了 `\newsavebox`, `\sbox` 和 `\usebox` 命令。这里 `\comname` 就是用 `\newsavebox` 命令提前创建的盒子名称, 其中放的是第一个参数值, 即注释人的姓名。利用这个新的定义, 注释的新样子如下:

Comment 3

In this form, every comment is assigned a sequential number after the word Comment. The comment text appears as before, while the name of the commenter is entered as the environment argument and is placed at the right of the last line. Tom William

要实现不只一个参数值的环境, 步骤同这里是完全一样的, 因此这里不再讲述。

7.4.3 有一个可省参数值的环境

同命令一样, 环境也可以有一个可省参数值。还是考虑上面那个例子, 如果我们觉得绝大多数注释都是由 Tom William 给出的, 这样我们就可以把定义的第一行改为

```
\renewenvironment{comment}[1][Tom William]{..}{..}
```

因此只有当注释人是其他人时才需要指定。

```

\begin{comment}[Alice] More than ...
... appropriate. \end{comment}

```

结果为

Comment 4

More than one person may want to make a comment, but perhaps one person makes more than others do. An optional argument for the name is then appropriate. Alice

7.5 对用户定义结构的解释

本节包含了用户定义 L^AT_EX 结构的创建和使用方面的一些一般性注解。这没有严格的规则, 只是阐述了许多 L^AT_EX 高手们根据自己的经验而得到的一些想法。每个用户都可以依据自己的体会, 总结出自己的实用方法。

7.5.1 保存用户定义的结构

用户创建自己定义的结构, 可以简化大量的工作。对于经常使用的命令和文本, 应利用 `\newsavebox`, `\newcommand` 或 `\newenvironment` 定义把它们单独写到一个文件中。这个文件可以用 `\input` 命令读入, 从而在其他文档文件中也可以使用。

随着时间的推移, 就会积累出大量的结构。若把它们全部放到一个文件中就会减慢处理过程, 而且跟踪内容和命令名称也变得非常困难。因此我们推荐用户结构应根据应用领域保存在许多独立的文件中。

7.5.2 缩写结构

用户定义命令的一个简单应用就是缩短 L^AT_EX 结构的名称。例如, 利用

```
\newcommand{\be}{\begin{enumerate}}
\newcommand{\ee}{\end{enumerate}}
```

只要输入 `\be` 就可以开始一个 L^AT_EX 环境 `enumerate`, 而 `\ee` 用来结束这个环境。

这样的缩写可以显著地避免许多输入, 但它并不适合于做为命令大量保存起来, 因为容易忘记命令名称背后所包含的意义。L^AT_EX 的作者 Leslie Lamport 已经精心选择了命令或环境的名称, 以明白地表示它们的功能。这种明确完整的命名, 要比不明了的任何缩写容易记住。但是可以在单个文档内容中使用某些缩写, 只要不太多就可以。具体要视用户的输入能力来确定了。

7.5.3 命令和计数器的名称相同

在前面的例子中, 一些特定的命令或环境的应用中使用了许多计数器, 例如 166 页上 `\myfootnote` 命令中的 `myfn`, 171 页上 `comment` 环境中的 `com`。在这些情形中, 计数器与对应的命令或环境有不同的名称。但并不是一定要这样, 计数器可以具有一个与命令或环境相同的名称。

例如, 在 166 页上, 我们可以将 `myfn` 计数器命名为 `myfn`。

↓

通常内部的结构已经被定义了。然而，这并不是必需的。用户定义中可以包含后面定义的其他用户结构。重要的是当第一次调用命令之前其他结构已经有了定义。

例如，下面是一个正常的定义顺序：

```
\newcommand{\A}{定义 A}
\newcommand{\B}{定义 B}
\newcommand{\C}{\A \B}
```

这里 \C 是用 \A 和 \B 定义的。然而，也可以像下面这样输入：

```
\newcommand{\C}{\A \B}
没有调用 \C 的普通文本
\newcommand{\A}{定义 A}
\newcommand{\B}{定义 B}
其他文本，可以随意调用 \A, \B 和 \C.
```

7.5.6 传递参数值

在命令和环境定义中的哑元参数值可以用作定义中其他命令的参数值。例如，如果命令 \A 和 \B 每个都有一个参数值，命令定义

```
\newcommand{\C}[3]{\A{#1}#2\B{#3}}
```

是可以接受的。这里 \C 的第一个和第三个参数值被传递给命令 \A 和 \B，只有第二个参数值直接进入定义。可以把直接参数值和传递参数值进行任意的组合。下面是一个比较具体的例子：

```
\newcommand{\sumvec}[4]{\anvec{#3}{#4} = #1_1+#2_1,\ldots,
#1_#4+#2_#4}
```

这样调用 `\sumvec xyzn` 就得到 $z_1, \dots, z_n = x_1 + y_1, \dots, x_n + y_n$ ，这里 \anvec 就是在 7.3.2 节中定义的命令。

7.5.7 嵌套定义

用户定义可以彼此嵌套，类似于下面这样的结构

```
\newcommand{\外部命令名}{\{\newcommand{\内部命令名}...\}}
```

是可以的。根据 174 页上关于定义作用范围的说明，用名称 {\内部命令名} 定义的命令只在命令 {\外部命令名} 内部才被知道和有效。虽然 TeX 的宏利用了大量的嵌套命令定义，以限制临时性命令的寿命，但是我们并不推荐过分地使用 L^AT_EX 定义嵌套，因为这样太容易造成括号不匹配。忘记一个括号匹配，将会在第二次调用外部命令时给出错误信息，因为这时还残留第一次调用时一部分内部定义。但是我们还是给出下面这个例子：

```
\newcommand{\twentylove}
{\{\newcommand{\fivelove}
{\{\newcommand{\onelove}
{I love \LaTeX!}%
\onelove\ \onelove\ \onelove\ \onelove\ \onelove}}}}
```

```
\fivelove\\ \fivelove\\ \fivelove\\ \fivelove\\ \fivelove}}
```

那么现在 My opinion of \LaTeX:\\ \twentylove 结果为:

My opinion of L^AT_EX:

I love L^AT_EX! I love L^AT_EX! I love L^AT_EX! I love L^AT_EX! I love L^AT_EX!

I love L^AT_EX! I love L^AT_EX! I love L^AT_EX! I love L^AT_EX! I love L^AT_EX!

I love L^AT_EX! I love L^AT_EX! I love L^AT_EX! I love L^AT_EX! I love L^AT_EX!

I love L^AT_EX! I love L^AT_EX! I love L^AT_EX! I love L^AT_EX! I love L^AT_EX!

在定义中像上面那样把行缩进, 可以帮助跟踪嵌套层次; 同一层次中的每一行不考虑括号的前提下开始于同一列。

如果内部和外部定义中都有参数值, 那么内外表示哑元参数值的符号必须不一样。内层定义的符号是 ##1 ... ##9, 而外层定义的符号 #1 ... #9。例如,

```
\newcommand{\thing}[1]{\newcommand{\color}[2]{The ##1 is ##2.}
\color{#1}{red} \color{#1}{green} \color{#1}{blue}}}
```

那么 The colors of the objects are\\

```
\thing{dress}\\ \thing{book}\\ \thing{car} 结果为
```

The colors of the objects are

The dress is red. The dress is green. The dress is blue.

The book is red. The book is green. The book is blue.

The car is red. The car is green. The car is blue.

如果采用 7.5.5 节中介绍的分开定义和调用命令的方式, 那么就比较容易接受:

```
\newcommand{\thing[1]}{\color{#1}{red} \color{#1}{green}
\color{#1}{blue}}
\newcommand{\color}[2]{The #1 is #2.}
```

7.5.8 不期望的空格

有时候用户定义结构中会出现不期望的空格, 或者出现比需要多的空格。这几乎总是由定义中的空白或新行造成的, 在定义中包含空格只是为了提高源文本可读性, 当结构被调用时它们可能被解释成空白。

例如, 如果在 166 页上的 \myfootnote 定义中去掉第一行的 % 字符, 那么就会在此处加入一个分行符, 从而转化成空白。这个空白要插入在接受脚注标记的前面单词与调用 \footnote 命令实际生成标记之间, 结果就会使得标记同单词分开。

在此处我们想提醒读者许多 L^AT_EX 命令是不可见的, 即在调用它们的地方不会生成任何文本。如果在这样不可见命令与周围命令之间有空格, 那么就可能出现两个空格。

```
For example \rule{0pt}{0pt} produces
```

结果为 ‘For example produces’, 有一个两倍单词间距。没有参数值的不可见命令不会导致这种问题, 因为跟在命令后面的空格被当做终止符, 从而消失了。而且, 下面的 L^AT_EX 命令和环境总是去掉后续空格, 即使它们有参数值:

<code>\pagebreak</code>	<code>\linebreak</code>	<code>\label</code>	<code>\glossary</code>	<code>\vspace</code>	<code>figure</code>
<code>\nopagebreak</code>	<code>\nolinebreak</code>	<code>\index</code>	<code>\marginpar</code>		<code>table</code>

7.5.9 最后两个例子

通常的 `description` 环境 (4.3.3 节) 把标签设成黑体, 所有接在第一行后面的行都缩进一个固定长度。为了描述计算机代码中的命令名, 可能更希望用打字机字体 (`\texttt` 命令或者 `\ttfamily` 声明) 显示标签, 后面的行缩进量等于最长标签的长度。为此我们现在定义一个新环境 `ttscript`, 其使用方法为

```
\begin{ttscript}{取样文本} 列表文本 \end{ttscript}
```

参数值 取样文本 在使用 `\texttt` 命令时的宽度等于左边缩进量。列表文本 由 `\item`[标签文本] 命令后接相应的解释性文本组成, 其中每行相对于左边界缩进 取样文本 的宽度。标签文本 在标签盒子中是用 `\texttt` 左对齐显示的。 `ttscript` 环境的定义为

```
\newenvironment{ttscript}[1]{%
  \begin{list}{}{%
    \settowidth{\labelwidth}{\texttt{#1}}
    \setlength{\leftmargin}{\labelwidth}
    \addtolength{\leftmargin}{\labelsep}
    \setlength{\parsep}{0.5ex plus0.2ex minus0.2ex}
    \setlength{\itemsep}{0.3ex}
    \renewcommand{\makelabel}[1]{\texttt{##1\hfill}}}
  \end{list}}
```

这也很容易把标签的字体改成其他字体样式, 重新给这个结构起个名字, 把它保存起来以供一般性应用。

对上面定义中的前七行的理解应该没有任何问题。环境的定义包含一个参数值, 它被如下传递:

```
\settowidth{\labelwidth}{\texttt{#1}}
```

这样就设置了标签的宽度。接着设置左边界的宽度:

```
\setlength{\leftmargin}{\labelwidth}
```

这样其值与标签 `\labelwidth` 是一样的, 最后

```
\addtolength{\leftmargin}{\labelsep}
```

在左页边界中增加了标签分隔 `\labelsep` 的量。

第六行和第七行把 `\parsep` 和 `\itemsep` 设置成适当的值, 用户可以按需要进行改变。如果必要的话, 也可以包含其他的列表参数。

最后一行重定义了 `\makelabel`, 这里需要加以解释。在 4.4.1 节中简要介绍了这一命令。它只在 `list` 环境中才有效, 每当调用 `\item` 时就用它生成实际的标签。它通常的参数值就是 `\item` 命令的可省参数值。利用上面的重定义, 现在的标签是包括字体命令 `\texttt` 后接 `\hfill`, 这样在标签盒子中是左对齐的。由于重定义是嵌套在 `ttscript` 环境定义中, 因此它的哑元参数值应是 `##1`, 而不是 `#1`, 在上一节中有解释。

例如, 给出如下文本:

```
\begin{ttscript}{description}
  \item[list] environments are useful for . . .
  \item[enumerate] environments number . . .
  \item[itemize] environments mark the . . .
  \item[description] environments label . . .
\end{ttscript}
```

那么我们可以得到如下关于 L^AT_EX 列表环境的描述:

list	environments are useful for generating lists in which the various items are set off from one another for greater clarity;
enumerate	environments number their items consecutively, starting at 1;
itemize	environments mark the various items with a distinguishing symbol, often a bullet •;
description	environments label the items with some text in bold face type, for greater stress.

用户定义的命令和环境可以有多达 9 个参数值。一个结构拥有的参数值越多, 其弹性就越大。另一方面, 调用命令时就变得很复杂和冗长, 因为参数的数目和顺序都必须严格与定义一致。

在 4.4.4 节中 72 页上的用户定义示例环境 `figlist` 中没有参数值。当调用它时, 每条 `\item` 命令生成 **Figure 1:**, **Figure 2:** 等等标签。而且文本相对于包围文本左边界缩进 2.6cm, 相对于右边界缩进 2cm。在列表声明中的类似于 `\labelsep`, `\parsep` 和 `\itemsep` 等其他列表参数取的是固定值。然而, 进行了下述定义后,

```
\newcounter{itemnum} \newlength{\addnum}
\newenvironment{genlist}[8]
{\begin{list}{\textbf{#1 \arabic{itemnum}:}}
{\usecounter{itemnum}
 \settowidth{\labelwidth}{\textbf{#1}}
 \settowidth{\addnum}{\textbf{\ \arabic{itemnum}: }}
 \addtolength{\labelwidth}{\addnum}
 \setlength{\labelsep}{#2}
 \setlength{\leftmargin}{\labelwidth}
 \addtolength{\leftmargin}{\labelsep}
 \setlength{\rightmargin}{#3}
 \setlength{\listparindent}{#4}
 \setlength{\parsep}{#5}
 \setlength{\itemsep}{#6}
 \setlength{\topsep}{#7#8}}
```

```
{\end{list}}
```

就能生成一个广义列表，其中第一个参数确定每次调用 `\item` 命令时与顺序编号显示在一起的共同项名称。左边的缩进设成项名称宽度加上 `\labelsep`，后者由第二个参数值给定。后面五个参数值确定各种列表参数值的长度。最后一个参数值为指定环境中文本的字体的声明。这个新环境的语法为：

```
\begin{genlist}{项名称}{标签间距}{右边距}{列表段落缩进}
  {段落间距}{项间距}{顶部间距}{字体样式}
  \item 文本
  ... ..
  \item 文本
\end{genlist}
```

当如下调用时，

```
\begin{genlist}{Sample}{4mm}{1cm}{0pt}{1ex plus0.5ex}
  {0pt}{0pt}{\slshape}
  \item no value
  \item The enclosed coupon is worth a value of \$2.00 towards
    the purchase of your next order.
\end{genlist}
```

结果为：

Sample 1: *no value*

Sample 2: *The enclosed coupon is worth a value of \$2.00 towards the purchase of your next order.*

在这个例子中，每个长度项都必须有单位。当然也可以在定义中直接包含单位，这样在调用时只需给出数值就可以了。同样竖直距离中的弹性长度也可以用如下定义中的算法给出：

```
\setlength{\parsep}{#5ex plus0.3#5ex minus0.5#5ex}
```

这里要求第五个参数值为纯粹的数字。如果其值为 2，`\parsep` 就会等于 `2ex plus0.6ex minus1ex`。这里还要提出的是，一定在要简化项数与记住其含义的复杂度之间取得折衷，特别是如果各种不同长度参数值具有不同的大小和算法的时候更要如此。

第八章 高级功能

前面几章主要讲述了如何用 \LaTeX 进行文本处理, 而在 \LaTeX 中还有一些功能, 对于高效地创建长而复杂的文档是相当重要的。这部分内容包括把一个文档分成几个文件, 文档中不同部分的选择处理, 章节、插图和公式的交叉引用, 自动生成参考文献、索引和汇总, 处理不同的字体集合, 准备报告材料和编辑书信。另外, 我们在本章中还介绍了外部图形的插入方法。

8.1 处理文档的各个部分

我们已经多次指出, \LaTeX 文档由导言和实际的文本两部分组成。对于较短的文档(如初学者通常处理的文档就是这样的情形), 可以用文本编辑器输入到单个文件中, 而且可以在第一次显示之后再进行修改。当用户积累了足够的经验和信心后, \LaTeX 文档就可能在长度上急剧膨胀, 甚至有可能生成长达 1000 页的宏篇巨著。

理论上这样长的文档是可以保存在一个文件中的, 虽然这会使得整个操作变得非常笨拙。对于长文件, 文本编辑器的功能大打折扣, 甚至有些编辑器根本就处理不了这么长的文件, 而且 \LaTeX 处理起来也会相应地花费很多时间。为此, 我们希望能有一种方法, 把作品分成几个文件, 在处理时由 \LaTeX 把它们合并起来, 或者有选择的编译处理某几个文件。

8.1.1 $\backslash\text{input}$ 命令

可以用下而这条命令把另一个文件的内容读进 \LaTeX 文档:

```
 $\backslash\text{input}\{\text{文件名}\}$ 
```

这里另一个文件的名称是 文件名.tex. 注意这里只需要指定另一个文件的基本名, 不需要加扩展名 .tex. 在 \LaTeX 处理过程中, 在第二个文件中的文本将会被包含进来, 放到第一个文件调用 $\backslash\text{input}$ 命令的地方。

$\backslash\text{input}$ 命令的结果与直接把文件 文件名.tex 中的内容输入到文档文件中该处是完全一样的。这条命令可以放在文档的任何地方, 既可以放在导言中, 也可以放在正文部分。

由于 $\backslash\text{input}$ 命令可以放在导言中, 因此可以把整个导言本身放到单独一个文件中。这样 \LaTeX 实际处理的文件甚至可以只包含命令 $\backslash\text{begin}\{\text{document}\}\dots\backslash\text{end}\{\text{document}\}$, 中间有很多 $\backslash\text{input}$ 命令。当有一系列的文档使用相同的导言时, 把导言放在单独一个文件中就是非常合理的。这样即使修改了导言, 也不必在所有文档中进行另外的修改。可以为各种不同类型的处理提供不同的导言文件, 然后用 $\backslash\text{input}\{\text{处理类型}\}$ 选择。

一个用 $\backslash\text{input}$ 命令读入的文件中也可以包含 $\backslash\text{input}$ 命令。嵌套深度只受计算机能力的限制。

为了得到所有读入文件的清单，可以把命令

```
\listfiles
```

放在导言中。当处理完毕，这个清单会同时出现在计算机显示器上和抄本文件中。版本号和其他的上载信息也会显示出来。这提供了一种检查到底有哪些文件被输入的方法。详情请见 C.2.9 节。

8.1.2 `\include` 命令

把文档分成几个文件，对于输入和编辑来说是比较实用的，但是当通过 `\input` 命令把它们合并起来后，处理的仍是整个文档。这样即使在一个文件中进行了很小的改动，所有的文件都要被重新读入和处理。因此我们希望能够提供一种方法，可以只重新处理被修改的那个文件。

一个比较粗糙的方法是写一个临时性文件，只包含导言（无论如何，它都需要被读入）和一条 `\input` 命令读入那个特定的文件。这种方法的缺陷是所有页码、章节、插图和公式等的自动编号都是从 1 开始的，因为所有得自前面的处理信息都没有了。而且，来自于其他文件中的交叉引用这时也行不通。

更好的方法是用 \LaTeX 命令

```
\include{ 文件名 }
```

这条命令只能用于文档的正文部分，另外命令

```
\includeonly{ 文件清单 }
```

只能位于导言中，文件名包含了要被读入的文件。而文件清单列出的则是当前正在处理的文件。文件名之间用逗号分开，不需要加后缀 `.tex`。这两条命令要结合使用。

如果文件清单中列出的文件恰是某条 `\include` 命令的参数，或者在导言中没有使用 `\includeonly` 命令，那么每条 `\include{文件名}` 命令就等价于

```
\clearpage \input{ 文件名 } \clearpage
```

然而，如果文件名不包含在文件清单中，`\include` 等价于 `\clearpage`，其中内容并不被读入。

`\include` 命令要比 `\input` 命令少一些普适性，因为它总是在新页上开始。因此文档应该在开始新页的地方分成文件，比如章与章之间。另一个局限性是 `\include` 命令不可以嵌套：它只能位于主处理文件中。然而，`\input` 命令可以出现在 `\include` 进来的文件中。

`\include` 命令的主要好处就是页面、章节和公式编号的附加信息可以由 `\includeonly` 命令提供，因此选择处理时这些计数器的值也是正确的。来自于其他文件中交叉引用信息也是可用的，因此 `\ref` 和 `\pageref` 命令 (8.3.1 节) 等也能生成正确的结果。所有这些值是由前面一次完整处理（用 \LaTeX 编译文件）确定下来的。

如果对被选择处理的文件进行了修改，导致页码的增加或减少，后面的文件也需要被重新处理以校正页码。如果增加或去掉了某些章节，或者公式、脚注和插图等的编号发生了变化，也需要进行同样的操作。

例如，假设文件三在第 17 页上结束，而经过选择处理后，它现在长度扩展到了第

22 页。但是后接的文件四 还是从第 18 页开始, 所有后面的其他文件也都具有自己的起始页码。如果文件四 被选择处理, 那么它将会得到正确的起始页码, 即根据修改后的文件三 保存的信息, 确定现在是从第 23 页开始。其他依次类推。然而, 如果在文件三 后选择处理的是文件六, 那么它将得到来自于文件五 的起始页码, 而这个页码还没有修正, 因此这之间会差 5 页。对其他结构、计数器也有同样的问题。只有当文件按正确的顺序进行了重新处理, 才能保证它们取得正确的值。

虽然有一些限制, 如果要处理长文档, `\include` 命令还是相当有用的, 它可以节省非常可观的用机时间。长文档在输入和编辑时通常要分很多步。`\include` 命令可以使得在很短时间内有选择地重新处理改动之处, 即使编号系统工作不正常也可以。随后进行一次完整的处理, 即去掉导言中的 `\includeonly` 命令就可以做到这一点。

用 `\include` 读入的文件中不能包含 `\newcounter` 声明。这并不是一个过分的限制, 因为通常就把这些声明放在导言中。

如果要写作一本书, 其每一章放在单独的文件中, 名称分别为 `chap1.tex`, `chap2.tex`, ...。那么处理文件本身就应包含如下文本:

```
\documentclass{book}
. . . . .
\includeonly{...}
\begin{document}
\frontmatter \include{toc}
\mainmatter
\include{chap1} . . . \include{chap8} . . .
\backmatter \printindex
\end{document}
```

这里 `toc.tex` 文件就是由如下文本组成:

```
\setcounter{page}{7}
\tableofcontents \listoftables \listoffigures
```

通过在 `\includeonly` 命令中填加适当的项, 就可以有选择地处理各章。例如, 在导言中加入 `\includeonly{toc, chap8}` 就可以只处理目录表和第 8 章。

8.1.3 终端输入和输出

有的时候希望在处理过程中 \LaTeX 能在计算机终端上显示出一条消息。这可以用如下命令来做到:

```
\typeout{信息}
```

这里的 信息 就代表要显示在计算机屏幕上的文本。当 \LaTeX 处理到这条命令时, 就会显示出这段文本。同时, 消息也写入到 `.log` 文件中。

如果 信息 中包含用户定义的命令, 那么就对它进行解释, 并把翻译后的结果显示在屏幕上。对 \LaTeX 命令也要做同样的事情。如果命令 (无论是用户定义的, 还是 \LaTeX 命令) 并不是可显示的, 那么这会造成可怕的后果。要显示命令名, 就在命令的前面加上

`\protect` 命令。

命令

`\typein[\命令名]{信息}`

也会把信息显示在终端上,但是它会等待用户从键盘上输入一行文本,用回车表示输入结束。如果没有可省参数值 `\命令名`,那么这文本就直接插入在处理过程中。举个例子,如果我们想重复利用一封信的文本,而写上几个不同的地址,只要每次从键盘上输入不同信息就可以了。假设文本是如下组成的:

Dear `\typein{Name:}\` ...

那么就会在屏幕上显示:

Name:

`\@typein=`

此时,我们可以输入收信人的姓名。如果的接连几次处理中输入了 'George', 'Fred' 和 'Mary',那么结果就是同样的信件内容,只是称呼不一样,它们是 'Dear George', 'Dear Fred' 和 'Dear Mary'。

如果 `\typein` 命令包含可省参数值 `\命令名`,那么就认为它等价于

`\typeout{信息}\newcommand{\命令名}{输入的定义}`

这样就会交互式地把定义保存在名称为 `\命令名` 的命令中,可以在文档的其他部分同其他 L^AT_EX 命令一样调用和执行这条命令。

对 L^AT_EX 方法有了一定经验后,就会发现利用 `\typein` 命令进行交互处理是可行的。例如,如果导言中包含

`\typein[\files]{Which files?}`

`\includeonly{\files}`

那么就会在屏幕上显示如下信息:

Which files?

`\files=`

L^AT_EX 等待用户输入一个或多个要处理的文件名(中间用逗号分开)。这就避免了每次必须用编辑器修改主处理文件。

当一封礼仪信件要送给不同的收信人时,也可以采用类似的过程。我们可以交互式地输入收信人的姓名、地址,甚至包括称呼。整封信由 L^AT_EX 用这种方法处理,用户从键盘上输入各项信息。

警告: `\typein` 命令不能用作其他 L^AT_EX 命令的参数值!然而它可以出现在类似于 minipage 这样的环境中。

8.2 在 L^AT_EX 中包含 T_EX 命令

T_EX 系统在应用上的快速扩展主要归功于 L^AT_EX 的实用性,因此有些用户可能根本不知道 T_EX。他们认为 L^AT_EX 就是一个单独可执行的程序,是与 T_EX 并存的。实际上, L^AT_EX 只是用户与 T_EX 处理程序之间的一个界面,它显著简化了 T_EX 操作。它把输入的逻辑结构翻译成起作用的 T_EX 命令,在内部通过调用 T_EX 来处理它们。

因此可以在 \LaTeX 内部包含纯粹的 \TeX 命令。这理所当然地适合于所有 \TeX 原语 (Primitive) 命令。除了大约 300 条 \TeX 原语命令外, 另外还有 600 个左右的定义在 Plain \TeX 格式中的宏。 \LaTeX 和 Plain \TeX 之间的真正区别就在于前者上载的格式文件是 `latex.fmt` (或者 \LaTeX 2.09 中的 `lplain.fmt`), 而后面上载的是 \TeX 格式文件 `plain.fmt`。然而, 由于 \LaTeX 格式包含了 Plain \TeX 中绝大多数宏定义 (注意并不是全部), 因此在 \LaTeX 中几乎可以调用所有的 \TeX 宏。

那些不能用在 \LaTeX 中的 \TeX 宏, 主要是因为它们在 \LaTeX 中已对其定义进行了修改, 或者干脆就没有定义, 其主要包括:

1. **\TeX 制表命令** 在 \LaTeX 中不能使用下列 \TeX 制表命令, 因为它们已被 `tabbing` 环境及其相关命令取代。

```
\tabs      \tabset      \tabsdone    \cleartabs
\settabs   \tabalign   \+
```

2. **页面格式, 脚注和浮动对象** 下列 Plain \TeX 命令已被删除:

```
\advancepageno  \footstrut      \nopagenumbers  \pageno
\dosupereject   \headline       \normalbottom  \plainoutput
\endinsert      \makefootline    \pagebody     \topins
\folio          \makeheadline    \pagecontents \topinsert
\footline       \midinsert       \pageinsert   \vfootnote
```

在 \LaTeX 中可以用 `\pagestyle`, 脚注命令, 以及 `figure` 和 `table` 环境等来达到同样的效果。

3. **\TeX 字体命令** 在 \LaTeX 中没有定义下列 Plain \TeX 字体命令:

```
\fivei   \fiverm   \fivesy   \fivebf
\seveni   \sevenbf  \sevensy   \teni    \oldstyle
```

利用字体以及字体尺寸命令很容易达到同样的效果。

4. **数学命令** 除了下面三条命令外的所有 \TeX 数学命令都可以用在 \LaTeX 中:

```
\eqalign   \eqalignno  \leqalignno
```

这三条命令已被 `eqnarray` 和 `eqnarray*` 环境代替。

5. **其他** 在 Plain \TeX 中的 `\beginsection` 命令已被 \LaTeX 中的章节命令取代。文档结束时, 用的是 `\end{document}` 命令, 而不是 `\end` 或者 `\bye`。在 \TeX 中的 `\centering` 命令已被 \LaTeX 中的同名命令所代替。在 \TeX 中有一名为 `\line` (写一行文本) 的命令, 它与 \LaTeX 的 `picture` 环境中的 `\line` (画一条直线) 冲突。在 \TeX 中 `\line` 命令所能完成的绝大多数任务, 在 \LaTeX 中用 `\center`, `\flushleft` 和 `\flushright` 都能完成。

\TeX 命令 `\magnification` 已完全被去掉了。需要时, 可以把放缩做为 DVI 驱动程序的选项, 这是完全合理的, 而且也非常有效。这样就不需要对 `.tex` 文件或 `.dvi` 文件进行修改。

尽管可以在 L^AT_EX 文档中使用 (Plain) T_EX 命令, 我们还是推荐, 只要有可能, 就尽量用高级命令。这是对可移植性和稳定性的最好保证。然而, 由于有很多精细的功能 (或技巧) 只能在 T_EX 的层次上得到, 因此禁止这种应用又是相当不明智的。

8.3 正文内的引用

在长文档中我们经常需要引用在其他地方进行了描述的章、节、表格和插图, 甚至是页码。同时也需要生成一个索引记录, 它是对整篇文档中特定关键词的引用。在利用计算机进行排版之前的年代里, 如此的交叉引用和索引意味着要耗费作者或秘书大量的工作时间。现在计算机可以为你担负该工作的绝大部分。

在以前岁月里, 引用前面文本部分的页码虽然是费事的, 但总是可行的。但是为了说明将要讲述的内容, 引用还没有写出来的部分就只能局限于章节编号了, 因为页码还不知道, 这时只好留下空白以备将来填上页码。

编写一本书, 通常是一个渐进的按部就班的工作。但是手稿可能根本就不是按顺序写的, 而且当初稿完成后, 基于作者新的考虑或者来自于审稿人富有见地的建议, 有可能对它进行很大的修改。修订、删除或插入某些章节, 都是完全有可能的, 更不用说有可能交换文本某些部分的顺序了。

L^AT_EX 把所有这些由于大改动造成的问题都变成了历史。无论作者进行了怎样的改动, 交叉引用和索引记录所需要的信息都被保存起来, 以供在正文中任何地方使用。

8.3.1 交叉引用

前面有多处地方已经提到过, 命令

```
\label{ 记号 }
```

用来给它所位于的文本中的点设置一个标记, 以便在其他地方引用这一位置。区别标记的符号是 记号 文本, 它可以是字母、数字和字符 (除了那些表示单个字符命令的特殊符号: `\` `#` `$` `%` `&` `~` `^` `_` `{` `}`) 的任意组合。

`\label` 命令被调用时所处的页码可以用下面命令来显示:

```
\pageref{ 记号 }
```

它可以位于文档的任何地方。

如果 `\label` 命令在章节命令后面给出, 或者位于 `equation`, `equarray` 或 `enumerate` 环境中, 或者是在 `figure` 或 `table` 环境中的 `\caption` 的参数值中, 命令

```
\ref{ 记号 }
```

就会用正确的格式显示出 记号 被定义处的章节、公式、插图、表格或枚举的编号。对于 `enumerate`, 显示出来的编号是 `\label` 所处的 `\item` 命令生成的编号。对于 `\newtheorem` 命令创建的定理类型结构, 如果 `\label` 命令出现在定理命令的文本中, 那么也可以引用它。例如, 在 73 页上 Hilbert 基定理的内容中用 `\label` 放上 记号 `hilbert-base`:

```
\begin{theorem}[Hibert]
    \label{hilbert-base}...\end{theorem}
```

因此输入文本

```
Theorem~\ref{hilbert-base} on page~\pageref{hilbert-base}
```

将得到输出 ‘Theorem 1 on page 73’。类似地，输入文本

```
for Table~\ref{budget95} on page~\pageref{budget95},  
see also Section~\ref{sec:figref}
```

结果为 ‘for Table 6.2 on page 155, see also Section 6.7.8’，因为那节包含 `\label{sec:figref}` 记号。

标记命令 `\label` 可以位于章节命令的参数值内，也可以位于该节正文中的其他地方。由相应的 `\ref` 命令显示的编号是 `\label` 出现的最内层那节的编号。为了避免可能出现的混淆，我们建议把 `\label` 命令放在被引用章节命令后面。

如果安装了 \LaTeX ，那么处理 `labl.st.tex` 文件就可以得到一张清单，其中包括所有的记号，记号转换后的结果，以及对应页码。

了解交叉引用信息的管理方式是相当有用的。实际上 `\label` 命令就是把记号文本连同相应计数器的值和当前页码写到一个辅助文件中，这个文件的基本名与正在处理的文档文件相同，后缀为 `.aux`。`\ref` 和 `\pageref` 命令就是从这个 `.aux` 文件中得到相关信息，这个文件是由 `\begin{document}` 命令读入的。在创建目录表时出现的情形，这里也会发生，即在第一次运行过程中，`.aux` 文件并不存在，因此不会输出任何交叉引用信息；而只是收集信息，并在第一次运行结束时写入一个新的 `.aux` 文件中。如果辅助文件已发生了改变，需要再运行一次，在本次处理结束时会给出一条警告信息。

8.3.2 参考文献的引用

在 4.3.6 节已讲述了参考文献的创建及对它的引用，这里重复讲述，只是为了给出完整的对交叉引用的讨论。参考文献是如下生成的：

```
\begin{thebibliography}{ 标签样本 }  
  \bibitem[ 标签一 ]{ 关键词一 } 文本条目一  
  \bibitem[ 标签二 ]{ 关键词二 } 文本条目二  
  . . . . .  
\end{bibliography}
```

在 4.3.6 节解释了参数值的意义，这里只解释一下引用关键词，不再重复其他各项。标记关键词扮演了 `\label` 命令中与记号相同的角色。而且它可以是字母、数字和字符的任意组合，只是不能用逗号。在正文中对参考文献的引用是用如下命令给出的：

```
\cite[ 附加信息 ]{ 关键词 }
```

它把相应 `\bibitem` 命令的标签放在中括号内。输入：

```
For additional information about \LaTeX\ and \TeX\  
see~\cite{lamport} and \cite{knuth, knuth:a}.
```

那么结合在 4.3.6 节的参考文献样例，就会得到如下输出：

For additional information about \LaTeX and \TeX see [1] and [6,6a].

引用标记 `lamport`，`knuth` 和 `knuth:a` 被转化成相应的参考文献中的引用标签。

如果在 `\cite` 命令中包含了可省参数值 附加信息, 那么这块文本就会加在标签的后面, 但仍然是在中括号内。

The creation of a bibliographic database is described in
`\cite[Appendix B]{lamport}`, while the program `\BibTeX`
 itself is explained in `\cite[pages 74,75]{lamport}`.

The creation of a bibliographic database is described in [1, Appendix B], while the program `BIBTEX` itself is explained in [1, pages 74,75].

在 `thebibliography` 环境中作者可以利用 `\bibitem` 命令逐项建立参考文献。另外还有一个单独的程序 `BIBTEX`, 它从一个或多个参考文献数据库中搜索那些出现在 `\cite` 命令中的 关键词 标记, 从而自动生成参考文献。这里的关键词必须与数据库中的一致。关于 `BIBTEX` 程序的使用, 请参考附录 B 的介绍。

8.3.3 索引记录

`LATEX` 并不会像处理目录表那样自动生成一个索引记录, 但它可以帮助作者建立关键词和页码的索引。

索引记录是用下面的环境生成的:

`\begin{theindex}` 索引条目 `\end{theindex}`

这个环境切换到两列页面格式, 它具有一个活动标题 `INDEX` (或在相应语言中的翻译名称)。在索引记录第一页上有一个标题 `Index` (或在相应语言中的翻译名称), 它的尺寸在 `book` 和 `report` 文档类中与章标题的尺寸相同, 在 `article` 中同节标题的尺寸一样。(更精确地说, 实际显示出来的单词是包含在命令 `\indexname` 中, 可以重定义它, 以适合其他语种。) 每一项都是用命令

`\item` `\subitem` `\subsubitem` 和 `\indexspace`

后接关键词和相应的页码组成的。例如,

commands, 18	<code>\item</code> commands, 18
as environments, 42	<code>\subitem</code> as environments, 42
arguments, 19, 101	<code>\subitem</code> arguments, 19, 101
multiple, 103, 104	<code>\subsubitem</code> multiple, 103, 104
replacement symbol, 20	<code>\subsubitem</code> replacement symbol, 20
used as arguments for sectioning commands, 41, 42	<code>\subitem</code> used as arguments for sectioning commands, 41, 42
	<code>\indexspace</code>
displayed text, 21-32	<code>\item</code> displayed text, 21--32

如果文本项太长, 在一行中放不下, 就会断开, 接在下一行上, 并且向内缩进, 深度比其他行都大, 如上面例子中的 ‘used as argument for sectioning commands, 41, 42’。命令 `\indexspace` 在索引记录中插入一个空白行。

`theindex` 环境只是建立起索引记录的一个适当格式。其中的每一项, 包括页码, 都必须手工输入。然而, 在确定页码方面 `LATEX` 可以提供一些帮助。

可以在正文中任何地方调用下面这条命令

`\index{索引条目}`

这里 索引条目 可以是任何文本, 但它应是后面索引中的一项, 它可以是字母、数字和符号的任意组合, 甚至可以包含命令字符和空格。这也就是说命令也可以包含在 索引条目 中, 如 `\index{\section}`, `\index{\{}` 或 `\index{\%}`。即使不能用作参数值的命令 `\verb` 也可以包含进来。然而, 如果 索引条目 包含命令, 那么它就不能再做为其他命令的参数值。另一个限制就是, 如果在 索引条目 中有一个左大括号, 那就必须同时给出右大括号。因此不能用 `\index{\{}`, 但可以用 `\index{\{\}}`。

除非导言中包含下面这条命令, 否则所有 `\index` 命令都会被 L^AT_EX 忽略:

`\makeindex`

这一命令激活所有的 `\index` 命令, 并打开一个文件, 其基本名与文档文件相同, 后缀 `.idx`。现在 `\index` 命令就会向这个文件中写入 索引条目 和当前页码, 格式为:

`\indexentry{索引条目}{页码}`

对于最简单的情形, 可以输出 `.idx` 文件, 从而得到一张索引项与对应页码的清单。利用这张清单, 作者就可以用前面给出的那种方式生成 `theindex` 环境中的各项。这应该是文档编写已到了最后阶段时的工作, 否则页码的改变会导致可怕的更正工作量。

在 L^AT_EX 的安装文件中有一个叫 `idx.tex` 的文件, 利用它可以提高 `.idx` 文件的可读性。当处理 `idx.tex` 文件 (即调用 `latex idx`) 时, 会提示用户输入要列出来的 `.idx` 文件名:

```
*****
* Enter idx file's first Name. *
*****
```

当从键盘上输入了 `.idx` 文件的基本名后, 就会输出两列页面格式, 它由出现在 Page n 形式页标题下面的 索引条目 文本组成。从这个有格式的列表中再得不到其他的信息, 但它要比直接利用输出的 `.idx` 文件容易得多。

`\filename=`

即使导言中没有使用 `\makeindex` 命令 (从而使得 `\index` 命令无效), 在着手编写文档的时候也应包含 `\index` 命令, 这是一个能节省大量时间的好主意。当文档终稿确定后, 再包含 `\makeindex` 命令, 从而激活已有的 `\index` 命令, 最后就可以利用来自于 `.idx` 或 `idx.dvi` 文件的各项来组织 `theindex` 环境。不过话又说回来, 这仍旧是一件烦人的工作, 需要相当大的耐心和集中的精力, 因为 `\item`, `\subitem` 和 `\subsubitem` 项必须按字母顺序, 而 `.idx` 只是按在文档中的顺序给出了所需信息。

然而, 还是有一个更好的方法。有一个可用的程序, 它从 `.idx` 文件生成 `theindex` 环境, 就如同 B^IB_TE_X 程序生成参考文献一样。在下一节讲述这一工具。

在 L^AT_EX 安装文件中也包含一个宏包 `showidx`, 它会把 `\index` 命令中的索引项变成边注, 在它所处的那页边界从页顶开始显示。当浏览文档的初始版本, 以看看所有的索引项是否位于正确的页面或者是否生成了新增项时, 这一方法是相当有用的。这个宏包是用 `\usepackage` 命令调入的, 或者把它放在 `\documentstyle` 的选项列表中。

如果用了 `showidx` 宏包, 那么可以在导言中用 `\marginparwidth` 声明 (4.10.7 节) 增加边注宽度, 以满足边注显示的需要。不幸的是, 现在的书籍格式并不适合于进行这种演示。

8.3.4 汇总

所谓汇总，就是一类特殊的索引，它是术语和短语按字母顺序排列，连同其解释一起组成的。为了帮助建立一个汇总， \LaTeX 提供了命令

`\makeglossary` 放在导言中，
`\glossary{汇总条目}` 放在正文部分

这两条命令的作用方式同组织索引记录是一样的。当在导言中使用了命令 `\makeglossary` 时，就会把各项写到一个后缀为 `.glo` 的文件中。文件中来自于每条 `\glossary` 命令的条目格式为

`\glossaryentry{ 汇总条目 }{ 页码 }`

在 `.glo` 文件中的信息可以用来建立汇总。然而，对于汇总，并不存在与 `theindex` 环境等价的结构，因此推荐的方法是利用 `description` 环境 (4.3.3 节) 或者特殊的 `list` 环境 (4.4 节)。

8.4 MakeIndex——关键词处理器

如果使用了 `MakeIndex` 程序，那么就可以避免利用 `theindex` 环境生成索引记录的苦差事。这个程序是 Pehong Chen 在 Leslie Lamport 的帮助下做出的。我们这里只是给出其用法的简略描述。在随程序所带的文档中有详细的讲解。

`MakeIndex` 程序处理 `.idx` 文件，得到输出文件，其基本名与文档文件相同，但是后缀为 `.ind`，这个文件由完整的 `theindex` 环境组成。程序的调用方法为：

`makeindex 基本名 .idx` 或者简单地 `makeindex 基本名`

接着当给出 `\printindex` 命令时， \LaTeX 经过处理，就在该命令处给出索引，这条命令连同 `\see` 命令一起，都是在宏包 `makeidx.sty` 中定义的。因此要用这种方法得到索引记录，需要用 `\usepackage` 命令装载 `makeidx` 宏包。

`MakeIndex` 程序希望 `\index` 项是下列三种形式之一：

`\index{ 主条目 }`
`\index{ 主条目 ! 子条目 }`
`\index{ 主条目 ! 子条目 ! 子子条目 }`

每个主条目，子条目或者子子条目可以是除 `!`、`@` 和 `|` 字符外的任意组合。在这里惊叹号是各个条目之间的分隔符。如果 `\index` 命令中只有一个主条目，那么它就会成为 `\item` 命令中的文本。主条目将以字母顺序排列。

如果 `\index` 命令由主条目和子条目组成，那么子条目的文本将赋给相应主条目下面的 `\subitem` 命令。`\subitem` 的文本也要以字母顺序排列。同样地，子子条目的文本将会赋给子条目文本下面的 `\subsubitem` 命令，并以字母顺序排列。

主条目和子条目也可以包含特殊字符，甚至可以是在排序过程中会被忽略的 \LaTeX 命令。其标志是条目的形式为 `排序条目@显示条目`，这里 `排序条目` 是用来进行字母排序的，而 `显示条目` 是实际要输出的文本。本书的索引是用 `MakeIndex` 生成的，所有的命令名都做为源

文本出现, 在排序时忽略前面的 \ 字符。这些条目就是用类似于 `\index{put@{\verb=\put=}}` 的文本得到的。

条目也可以用字符序列 `|` (或 `|`) 结尾, 以标志页码范围的开始与结束。例如, 如果在第 126 页上有 `\index{picture!commands|}`, 在第 135 页上有 `\index{picture!commands|})`, 结果就是在主条目 'picture' 下面的子条目 'commands' 后面显示页码 126–135。

可以不在条目后面显示页码, 而代之以对另一条目的引用。例如, 利用

```
\index{space|see{blank}}
```

就会在索引中得到 'space, see blank'。(更准确地说, 是存贮在 `\seename` 命令中的文本显示在 *see* 的地方; 这样可以改变它以适应其他语言。)

对于 MakeIndex 程序, `!`, `@` 和 `|` 这三个字符因此具有特殊的作用。为了按原样以文本方式显示这三个字符, 不让它们发挥本来的作用, 需要在其前面加上引号符号 `"`。例如, `"!` 表示的就是一个惊叹号, 而不是项分隔符。

因此引号就成为第四个特殊符号, 必须用输入 `"` 得到本来的样子。然而, 在 MakeIndex 语法中有一条特殊的规则, 那就是在引号前面加上反斜杠, 就会认为它是命令的一部分, 因此在条目中就可以用 `\` 来得到德语重音 (如 `\index{Knappen, J\"org}`)。这条特殊的规则有时会导致另外的问题, 例如在本书中索引条目 `\!` 就必须输入为 `\\"!`。

可以把页码指定为不同的字体。例如, 本书的索引记录中, 黑体的页码用来表示命令第一次被解释或定义的地方。这是用如下形式的项得到的:

在第 11 页上有 `\index{blank}`, 在第 18 页上有 `\index{blank|bb}`

第二种情形中把该项的页码作为命令 `\bb` 的参数值。在 `theindex` 环境中的对应行变为

```
\item blank, 11, \bb{20}
```

这一方法只有当已经在导言中进行了 `\newcommand{\bb}[1]{\textbf{#1}}` 时才可行。注意: 在 `\index` 项中的竖线并不是排版失误, 在这种情况下用它来代替 `LATEX` 命令符号反斜杠。

也可以定义其他的页码样式, 如

```
\newcommand{\ii}[1]{\textit{#1}}
```

```
\newcommand{\zz}[1]{\textsl{#1}}
```

分别表示斜体和 slanted 字体。在 `\index` 命令中就要用 `!ii` 或 `!zz` 结束该项以进行相应的定义。

在 MakeIndex 程序中的字母排序通常是按照标准的 ASCII 码进行的, 首先是符号, 然后是数字, 最后是字母, 而且大写字母在小写字母前面。有许多选项可以改变这些规则。当调用程序时如何给出选项与计算机类型有关, 这里我们假设在选项字母前面加连字符表示选项, 如

```
makeindx -g -l 基本名
```

最重要的选项有:

-l 字母顺序: 排序时忽略空格;

-c 压缩空格: 同通常的 `LATEX` 一样, 忽略多于一个的空格或者前导空格。

-g 德语顺序: 根据德语顺序, 符号在字母前面, 小写字母在大写字母前面, 然后是数字; 而 "a, "o, "u 和 "s(在德语版的 L^AT_EX 中表示 ä, ö, ü 和 ß) 就当做它们是 ae, oe, ue 和 ss 处理, 这与标准德语中的用法一致。

-s 样式定义: 允许给出索引格式文件的名称, 以包含重定义的 MakeIndex 功能。

-s 选项读入一个索引样式文件, 这个文件由定义 MakeIndex 程序输入和输出的命令组成。例如, 可以用另外的字符来替换特殊字符 !, @, | 和 ", 这样新的字符具有原来的作用, 而原来的字符成为纯粹的文本。

样式定义文件由一串关键词属性对组成。属性可以是由在单引号内的一个字符组成 (例如 'z'), 也可以是在双引号内的字符串组成 (如 "a string")。最重要的关键词, 及其默认定义为:

```
quote    '""'    定义引号符号;
level    '!'     定义项分隔符号;
actual    '@'     定义词汇切换符号;
encap     '|'     定义页面格式中的虚命令符号。
```

有相当多的关键词用来定义复杂输出结构。这些关键词在随 MakeIndex 程序一起发行的文档中有讲解。

文件 makeindex.tex 包含了 Leslie Lamport 给出的一个简略手册 (其中没有提到样式定义)。另外, 在 DOS 版本的 emT_EX 中, MakeIndex 程序有两个版本, 一个名称为 makeidx.exe, 另一个名称为 mkidx32.exe。这两个程序的区别在于后者可以处理的索引大小要远远大于前者, 因此一般情况下, 我们就采用 mkidx32.exe 来处理 .idx 文件。

8.5 新字体选择框架 (NFSS)

在刚发明 T_EX 和 L^AT_EX 的时候, 可用的字体在数量上是很有限制的。正是由于这个原因, 在 L^AT_EX2.09 中定义字体所用的系统弹性很差, 因为我们需要改变它们的必要性实在不是很明显。高级字体命令 (如 \large 和 \bf) 同最终选定的外部字体名称之间的关联严格地固定在文件 lfonts.tex 中, 而这个文件就组合在 lplain 格式中。

到了今天, 可用的字体很多, 其中有些字体可以与标准计算机现代 (CM) 字体一起共用, 而其他一些则需要取代某些 CM 字体。例如, Cyrillic 字体须平行于 Latin 字体加进来, 但是要想在标准 L^AT_EX 尺寸命令下自动对它进行操作, 则是相当复杂的过程。(我们已经知道这是可以做到的!) 同样, 安装 PostScript 字体激活了对错综复杂的界面宏的调用, 这些宏是来自于 PostScript 驱动宏包, 但是它包含了一批 L^AT_EX 软件开发人员的大量心血。

在 L^AT_EX2.09 中另一个问题是字体样式和尺寸命令的行为 (4.1.5 节和 4.1.2 节)。字体声明 \rm, \bf, \sc, \sl, \it, \sf 和 \tt 中的每一个都激活与当前选定尺寸有关的特定的字体。这些声明中每一个都是排它的。因此 \bf\it 的组合实际上就与 \it 相同。在这个框架中无法选择黑斜字体。而且, 从 \tiny 到 \Huge 的尺寸声明都是自动切换到 \rm, 因此 \bf\large 并不生成所期望的黑体大号字体。最后, 这里使用的是声明, 而不

是有一个参数值的命令，这与 L^AT_EX 命令的基本哲学相矛盾。也就是说，为了强调单个词，输入 `\emph{single}` 就要比 `{\em single\}` 更合理些。（有经验的 L^AT_EX 用户可能会否认这一点，但这只是因为他们已完全习惯于原来的那一套。）

在 1989 年，Frank Mittelbach 和 Rainer Schöpf 为 L^AT_EX 提出了一个新字体选择框架 (NFSS)，并在 1990 年给出了一个初步的测试宏包。在 1993 年年中，第二个版本 (NFSS2) 问世，这一版本做了相当大的改进。在 1994 年 4 月正式发行的 L^AT_EX 2_ε 中，NFSS 已经在新标准中占稳了脚跟。新字体声明和命令在 4.1.3 节和 4.1.4 中已做了讲述。这里我们更仔细地讲解这个系统，给出一些低层次的字体选择命令。

8.5.1 NFSS 中的字体属性

按照 NFSS 框架，每个字符集可以根据如下五种属性分类：编码、族、序列、形状和尺寸，用户通过下面的命令来选择这些属性：

```
\fontencoding{ 编码 } \fontfamily{ 族 } \fontseries{ 权与宽度 }
\fontshape{ 形状 } 和 \fontsize{ 尺寸 } { 基线间距 }
```

编码属性是在 NFSS 第二版本中新出现的。它定义字体中字符的布局。在表 8.1 中给出了它的可能取值。我们一般不大可能需要在一片文档中改变编码，当然激活 Cyrillic 字体除外。这个功能就是为了允许程序开发者可以在系统中安装新字体而设计的。

表 8.1: NFSS 编码框架

编码	描述	字体样例
OT1	来自于 Knuth 的原始文本字体	cmr10
OT2	Washington 大学的 Cyrillic 字体	wncyr10
T1	Cork(DC) 字体	dcr10
OML	T _E X 数学字母字体	cmmi10
OMS	T _E X 数学符号字体	cmsy10
OMX	T _E X 数学扩展字体	cmex10
U	未知编码	—

在 `\fontfamily` 命令中的 族 参数值表示字体的一组基本属性 (或来源)。对于计算机现代字体，所有 serif 字体都属于 cmr 族。而族 cmss 包含所有 sans serif 字体，族 cmtt 包含所有打字机字体。一些特殊装饰性字体是它们所在族的唯一成员。在表 8.3 中根据族和其他属性列出了所有的 CM 字体。

注意：字体属性“族”与原来 T_EX 中同名概念之间没有任何关系。一个 T_EX 的族可以由在数学公式中做为普通文本、第一层和第二层下标所用的三种不同尺寸的字体组成。

在 `\fontseries` 中的参数值 权与宽度 表示字符的权 (= 粗细程度) 和宽度。它们由表 8.2 中所给的 1 到 4 个字母来定义。

`\fontseries{ 权与宽度 }` 的参数值是由对应于权的字母后接对应于宽度的字母组成。因此 `eb` 表示权为较黑，宽度为较松，而 `bx` 意味着权为黑，宽度为松。同任何非正常

表 8.2 NFSS序列属性

权类		宽度类		
超轻	ul	超紧	50%	uc
较轻	el	较紧	62.5%	ec
轻	l	紧	75%	c
半轻	sl	半紧	87.5%	sc
中间 (正常)	m	中间	100%	m
半黑	sb	半松	112.5%	sx
黑	b	松	125%	x
较黑	eb	较松	150%	ex
超黑	ub	超松	200%	ux

权或宽度组合时, 字母 m 可以忽略; 如果两者都是正常, 那么只要给出 m 就可以了。

在 `\fontshape` 中, 参数值 形状 是 n, it, sl 或 sc 字母组合中的一种, 分别表示正常 (直立)、斜体、slanted 或小体大写字母。

`\fontsize` 属性命令有两个参数值, 第一个参数值 尺寸 表示字体以点为单位的大小 (不显式地给出 pt 单位), 而第二个参数值 基线间距 是从一个基线到下一个基线的竖直距离。第二个参数值成为 `\baselineskip` 的新值 (3.2.4 节)。例如, `\fontsize{12}{15}` 选择 12pt 的字体尺寸, 行间距为 15pt. (第二个参数值可以给出单位, 例如 15pt, 但如果没有给出单位, 就认为单位是 pt.)

一旦五个属性都已设置好, 就可以用 `\selectfont` 命令选择字体。这里的新特征是各种属性彼此独立。改变其中一个, 并不会改变另一个。例如选择:

```
\fontfamily{cmr} \fontseries{bx} \fontshape{n} \fontsize{12}{15}
```

就会生成一种直立、黑体、松的罗马字体, 尺寸为 12pt, 行间距为 15pt. 如果接下来利用命令 `\fontfamily{cms}` 选择了 sans serif 字体, 那么属性中的权和宽度 (bx), 形状 (n), 尺寸 (12(15pt)) 在下一调用 `\selectfont` 命令时继续有效。

等价地, 可以利用下面这条命令来定义除尺寸外的所有属性, 并同时马上激活字体;

```
\usefont{代码}{族}{序列}{形状}
```

在表 8.3 (由 F. Mittelbach 和 R. Schöpf 提供) 中列出了根据 `\fontfamily`, `\fontseries` 和 `\fontshape` 属性对计算机现代字符集的分类。有相当多的属性组合并不对应任一 CM 字体, 这看起来好像是 NFSS 系统的一种缺陷, 但我们要知道, 这一设计是为将来考虑的。它也可以应用于正变得越来越普及的 PostScript 字体, 以开发其完整用途。

从形式上看, 我们可以任意设置属性的组合。然而, 可能不存在一种字体对应于所选择的属性。如果出现这种情况, 那么当调用 `\selectfont` 时, \LaTeX 会给出一条警告信息, 告诉你它用什么字体取代了所需字体。在 `\fontsize` 命令中的字体 尺寸 属性通常可以取 5, 6, 7, 8, 9, 10, 10.95, 12, 14.4, 17.28, 20.74, 但也可以是其他值。第二个参数值, 即 基线间距, 可以取任何值, 因为它并不是字体固有的性质。

表 8.3 计算机现代字体的属性

序列	形状	外部字体名称示例
计算机现代罗马字体 --(\fontfamily{cmr})		
m	n, it, sl, sc, u	cmr10, cmti10, cmsl10, cmcsc10, cmu10
bx	n, it, sl	cmbx10, cmbxti10, cmbxsl10
b	n	cmb10
计算机现代 Sans Serif 字体 --(\fontfamily{cmss})		
m	n, sl	cmss10, cmssi10
bx	n	cmssbx10
sbc	n	cmssdc10
计算机现代打字机字体 --(\fontfamily{cmtt})		
m	n, it, sl, sc	cmtt10, cmitt10, cmsl10, cmtcsc10

利用 `\begin{document}` 命令, L^AT_EX 给五种属性设置当前特定默认值。这通常就是标准编码 OT1, 族 cmr, 中间序列 m, 正常形状 n 和选择的基本尺寸。用户可以在导言中改变这些值, 或者用特殊选项把它们设置成不同的值, 例如当已经选定了一种 PostScript 字体的时候, 就有必要进行这种操作。

8.5.2 简化字体选择

属性命令 `\fontencoding`, `\fontfamily`, `\fontseries`, `\fontshape` 和 `\fontsize`, 以及 `\selectfont` 命令, 都是新字体选择框架中的基本工具。用户并不需要直接使用这些命令, 而可以利用列在 4.1.2 节和 4.1.3 节的高级声明。事实上, 类似于 `\itshape` 这样的字体声明就是定义为

```
\fontshape{it}\selectfont.
```

用来选择字体尺寸的高级命令有:

```

\tiny (5pt)   \normalsize (10pt)   \LARGE (17.28pt)
\scriptsize (7pt)   \large (12pt)   \huge (20.74pt)
\footnotesize (8pt)   \Large (14.4pt)   \Huge (24.88pt)
\small (9pt)
```

当在 `\documentclass` 命令中选择 10pt(默认值) 做为基本尺寸选项时, 上面命令相应的尺寸就是列在括号内的数值; 当选择了 11pt 或 12pt 时, 这些尺寸就会相应的放大。

族声明及对应的标准族属性值为

```
\rmfamily (cmr)   \sffamily (cmss)   \ttfamily (cmtt)
```

这分别相应于计算机现代字体族中的罗马、Sans Serif 和打字机字体 (E.2 节)。

序列声明及其初始值为:

```
\mdseries (m)      \bfseries (bx)
```

这就是说在标准 L^AT_EX 中只提供了中间和黑松两种序列属性。

最后, 形状声明和相应属性值为:

```
\upshape (n)      \itshape (it)
\slshape (sl)      \scshape (sc)
```

利用这些声明可以选择直立、slanted、斜体和小体大写字母。

注意对编码并没有高级声明。这是因为通常并不需要在文档中改变编码。但是, 如果要使用 Cyrillic 字体 (编码 OT2), 就需要利用与编码有关的命令。此时可以进行如下定义:

```
\newcommand{\cyr}{\fontencoding{OT2}\selectfont}
\newcommand{\lat}{\fontencoding{OT1}\selectfont}
```

这样就可以方便地来回切换了。

利用 \normalfont 命令, 可以随时把族、形状和序列属性值重设为标准值, 这也激活了当前尺寸的那种字体。

对于上面每种字体属性声明, 也存在着一个相应的字体命令 (4.1.4 节), 用来设置其参数值的字体。因此 \textit{text} 就与 {\itshape text} 差不多一样, 唯一的区别就在于命令中自动包含倾斜校正。这些命令的完整清单如下:

```
族:      \textrm  \textsf      \texttt
序列:    \textmd  \textbf
形状:    \textup  \textit      \textsl  \textsc
其他:    \emph    \textnormal
```

在 4.1.1 节中描述了 \emph 命令; \textnormal 把其参数值字体设为 \normalfont。

8.5.3 默认属性值

在前面一节中的字体属性声明并没有显式地设置它们的值, 而是用某种默认的命令来进行。因此 \itshape 的真正定义是:

```
\fontshape{\itdefault}\selectfont
```

可用的默认命令有:

```
族:      \rmdefault  \sfdefault  \ttdefault
序列:    \mddefault  \bfdefault
形状:    \updefault  \itdefault  \sldefault  \scdefault
```

当调用了 \normalfont 命令时, 需要定义标准属性值。它们是由如下四个默认值组成的:

```
\encodingdefault  \familydefault  \seriesdefault
\shapedefault
```

所有这一切都使得我们觉得好像高级命令与特定字体之间的联系是非常复杂的。实际上, 它确实提供了足够的弹性和模块化结构。作者只需要知道这三个族、两个序列和

四种形状是可以用的，而不必关心它们实际是什么。程序设计者用低级命令定义它们的默认值。

在 200 页上有一个例子，通过重定义三个族的默认值，说明如何利用 PostScript 字体取代所有四种标准字体。这样重定义比改变包括 `\normalfont` 在内的字体声明要简单得多。这些定义实际上要远比这里提到的复杂，然而默认命令确实就如这里所指出的那样简单。

8.5.4 定义字体命令

有很多命令可以用来定义新的字体声明和命令。这些命令主要是为 L^AT_EX 宏包开发者提供的，但也可以用在普通文档中。

```
\DeclareFixedFont{\命令}{编码}{族}{序列}{形状}{尺寸}
```

把 `\命令` 定义为一个选择具有指定属性的字体的声明。列出的所有属性都是严格固定的。它与 `\newfont` 命令基本等价，只是这里的字体由属性确定，而不是由名称确定。

```
\DeclareTextFontCommand{\命令}{字体指定}
```

定义 `\命令` 为一个字体命令，它按照 字体指定 设置其参数值。在内部就是用这条命令来定义所有类似于 `\textbf` 的命令，而定义 `\textbf` 时 字体指定 为 `\bfseries`。

```
\DeclareOldFontCommand{\命令}{文本指定}{数学指定}
```

定义 `\命令` 为按照 L^AT_EX2.09 方式可以用在数学模式中的字体声明。注意它是一个声明，不是一条命令。这对于定义与原来版本兼容命令是相当有用的，但应尽量避免使用。例如，`\it` 的定义为

```
\DeclareOldFontCommand{\it}{\normalfont\itshape}{\mathit}
```

8.5.5 数学字母表

已被激活的用于文本处理的字体对数学模式中的字符及其字体并没有作用，因为此时用的是特殊的数学符号字体。如果想使一个公式显示为黑体，那么就必须用 `\boldmath` 命令 (5.4.10 节)，该命令的作用持续到调用相反命令 `\unboldmath` 为止。这些声明都必须在数学模式外面给出。

在 NFSS 下也可以通过同样方式应用这些声明。然而，内部的数学字体选择命令为

```
\mathversion{变体名称}
```

这里的参数值 变体名称 通常就取 `normal` 和 `bold`。 `\boldmath` 和 `\unboldmath` 声明就是用这一命令定义的。

另一方面，可以在数学模式内部调用下列数学公式中的字体命令，把字母设置成特定的字体 (5.4.3 节)：

```
\mathrm \mathcal \mathnormal \mathbf \mathsf \mathit \mathtt
```

这些都是对参数值有作用的命令，而不是声明，这一点与 L^AT_EX2.09 中的不同。

新数学字体字母表也可以由用户定义。例如，为了定义 slanted 数学字体 `\mathsl`，可以用

```
\DeclareMathAlphabet{\mathsl}{OT1}{cmr}{m}{sl}
```

这就意味着新数学字体命令 `\mathsl` 选择族为 `cmr`, 权为 `m` 和形状为 `sl` 的字体, 在通常字体定义中, 这就是适当尺寸的 `cmsl` 字体。虽然在所有数学变体下, 都可以选择这种字体, 但是当 `\mathversion{bold}` 命令起作用时, 选择黑体字体就可能更恰当一些。为此可以在 `\mathsl` 定义中加入该选项:

```
\SetMathAlphabet{\mathsl}{bold}{OT1}{cmr}{bx}{sl}
```

这样就定义 `\mathsl` 为期望的只适用于 `bold` 变体, 权为 `bx` 的字体。对于普通的字体定义, 这就是当前尺寸的 `cmbxsl`。

可以用下面的命令创建新的数学变体:

```
\DeclareMathVersion{ 变体名称 }
```

属于它的字体是由对每个数学字母表或符号字体调用 `\Set...` 命令确定的。

8.5.6 数学符号字体

数学符号的定义方式与文本字符的定义方式完全不同, 数学符号拥有一个命令名 (如 `\alpha`), 它们可能来自于不同的字体, 根据不同的类型有不同的行为, 可以显示为不同的尺寸。在 $\text{\LaTeX}2.09$ 中, 符号名称固定为计算机现代数学字体, 但 NFSS 为选择其他 (或者代替) 符号字体提供了一个相当大的弹性。

用下面这条命令声明一个符号字体名称:

```
\DeclareSymbolFont{ 符号字体名称 }{ 编码 }{ 族 }{ 序列 }{ 形状 }
```

这样就把 符号字体名称 与给定的属性集联系起来。这一名称不是一条命令, 而是一个内部用来定义符号的标识。所选定的字体对所有变体都有效, 除非在其他变体中指定了同名的不同字体。

```
\SetSymbolFont{ 符号字体名称 }{ 变体 }{ 编码 }{ 族 }
{ 序列 }{ 形状 }
```

可以用来重新定义一种变体下的 符号字体名称。

标准 \LaTeX 安装中有如下声明:

```
\DeclareSymbolFont{operators}{OT1}{cmr}{m}{n}
\DeclareSymbolFont{letters}{DML}{cmm}{m}{it}
\DeclareSymbolFont{symbols}{OMS}{cmsy}{m}{n}
\DeclareSymbolFont{largesymbols}{OMX}{cmex}{m}{n}
```

这一串声明是在 \LaTeX 内部定义的, 这也是它们之所以重要的原因。

一旦定义了符号字体名称, 就可以用它来构造数学字母表和各种不同类型的符号。

```
\DeclareSymbolFontAlphabet{\数学字母表}{ 数学字体名称 }
```

把 `\数学字母表` 定义成基于内部名称 `数学字体名称` 的数学字母表。如果相应于这个数学字母表的字体合适属性已经存在, 那么这条命令就要优于命令 `\DeclareMathAlphabet`。

定义符号的主要命令是

```
\DeclareMathSymbol{\符号}{ 类型 }{ 符号字体名称 }{ 位置 }
```

这使得 `\符号` 显示在字体 `符号字体名称` 中处于给定位置的符号。这里 `位置` 是一个数, 可以用十进制 (如 10), 八进制 (如 '12) 或十六进制 (如 "0A) 表示。 `类型` 定义符号的功

能, 它可以取如下一个值:

<code>\mathord</code>	普通符号	<code>\mathop</code>	大运算符, 如 \sum
<code>\mathbin</code>	二元运算符, 如 \times	<code>\mathrel</code>	关系运算符, 如 \geq
<code>\mathopen</code>	左括号, 如 $\{$	<code>\mathclose</code>	右括号, 如 $\}$
<code>\mathpunct</code>	标点符号	<code>\mathalpha</code>	字母表字符

数学字母表命令只作用在类型为 `\mathalpha` 的符号上; 对于其他类型, 在给定的数学变体里, 在所有数学字母表中都生成同样的符号。

上面的数学字体声明中并没有指定尺寸。这是因为通常有四种尺寸可以使用, 具体与数学样式有关, 可看 5.6.2 节的解释。然而, 这些尺寸必须在某个地方指定。这是用如下命令完成的:

```
\DeclareMathSizes{正文}{数学文本}{上下标}{双重上下标}
```

这里的四个参数值都是数值, 给出尺寸的点数。当正文公式中字体的尺寸是正文 pt 时, `\textstyle` 就会是数学文本尺寸, `\scriptstyle` 是上下标尺寸, 而 `\scriptscriptstyle` 是双重上下标尺寸。例如,

```
\DeclareMathSizes{10}{10}{7}{5}
```

所有的 `\Declare...` 和 `\Set...` 命令都只能在导言中调用。

8.5.7 获取属性值

在程序设计中, 有时要利用属性的当前值。但是通常我们并不知道它们的具体值是多少, 这些属性值实际上保存在如下内部命令中:

```
\f@encoding \f@shape \tf@size
\f@family \f@size \sf@size
\f@series \f@baselineskip \ssf@size
```

注意不要直接改变这些命令的值。然而, 可以测试它们, 以确定它们是否具有某个值。由于其名称中都包含字符 `@`, 因此它们只能直接用在类或宏包文件中, 如果要在主文档文件中使用, 需要把它们包含在 `\makeatletter` 和 `\makeatother` 之间, 见 101 页上的用法。

8.5.8 定义 NFSS 中的字体

在 NFSS 中, 如果需要在文档中指定字体, 那么就必须先给出所需要的属性, 然后调用 `\selectfont`。这样确定的字体属性集合是如何与特定外部字体名称联系起来的呢? 方法是通过字体定义命令做到的, 它通常保存在扩展名为 `.def` 和 `.fd` 的文件中。

首先利用声明

```
\DeclareFontEncoding{编码}{正文集合}{数学集合}
```

建立一个叫 编码 的新编码属性; 无论何时选择这种编码中的一个字体, 就会执行 正文集合 中的命令, 以重新定义重音命令或其他与编码有关的事情; 同样地, 对于这种编码中的每个数学字母表都会调用并执行 数学集合 中的命令。也可以如下定义默认的 正文集合 和 数学集合:


```
\DeclareFontEncodingDefaults{ 正文集合 }{ 数学集合 }
```

可以用这条命令声明一般的文本和数学模式设置，而更具体的，在后面执行的设置是位于 `\DeclareFontEncoding` 中的。

如果对应于指定的属性，不存在任何字体，

```
\DeclareFontSubstitution{ 编码 }{ 族 }{ 序列 }{ 形状 }
```

声明了用来取代它的属性值；取代是按照从 形状, 序列, 然后 族 的顺序进行的；编码从不会被取代。如果这样还找不到对应字体，那么

```
\DeclareErrorFont{ 编码 }{ 族 }{ 序列 }{ 形状 }{ 尺寸 }
```

就确定出最终迫不得已时使用的字体。

指定编码框架中的一个新族是用如下命令建立的：

```
\DeclareFontFamily{ 编码 }{ 族 }{ 选项 }
```

这里的 选项 是一组命令，每当选择该族和编码中的一种字体时就会被执行。

把字体属性与外部字体名称关联的主要字体定义声明是：

```
\DeclareFontShape{ 编码 }{ 族 }{ 序列 }{ 形状 }{ 字体定义 }{ 选项 }
```

这里的 选项 是另外的命令，每当选择其中一种字体时就会执行它。

字体指定包含一系列尺寸 / 字体联系，每一个联系都是由一个尺寸部分、一个函数、一个可省参数和一个字体参数值组成。例如，

```
\DeclareFontShape{OT1}{cmr}{m}{n}
{
  <5> <6> <7> <8> <9> <10> <12> gen * cmr
    <10.95> cmr10
    <14.4> cmr12
    <17.28> <20.74> <24.88> cmr17}{}
```

就说明 `cmr` 族的中间序列、正常形状成员用外部字体 `cmr5 ... cmr12` 表示 5–12pt 的尺寸，用 `cmr10` 放大到 10.95 以接近 11pt 的尺寸，等等。如果指定的尺寸不存在，就会用特定限度内最接近的尺寸。

尺寸部分由一个或多个放在尖括号内的表示尺寸 (pt) 的数字组成。括号内也可以包含范围，如 `<-10>` 就表示所有小于 10pt 的尺寸，而 `<10-14>` 表示从 10pt 到小于 14pt 的尺寸，而 `<24->` 表示 24pt 或更高。可能的函数有：

(empty) (上面例子中的第二、三、四行) 上载指名的字体，并放缩到要求的点数尺寸；如

如果在字体名称前面有位于中括号内的可省参数值，那它就是一个额外的放缩因子：

```
<11> [.95] cmr10    就会上载 cmr10 并放缩到 11pt 的 95%；
```

`gen *` (上面例子中的第一行) 把点数尺寸加到字体参数值后，生成字体名称：

```
<12> gen * cmr      上载 cmr12；
```

`sub *` 用不同的字体代替，这种字体的属性在形式为 族/序列/形状 的字体参数值中给出；

```
<-> sub * cmtt/m/n    当没有字体具有所要求的属性时，最好用这种字体；
```

会在监视器和抄本文件中给出一条信息；

`subf *` 类似于空函数，但是如果上载了一种显式取代字体时会给出一条警告信息；

`fixed *` 以正常尺寸上载指定字体, 忽略尺寸部分; 如果给出了可省参数值, 那这个参数值就是字体要放缩到的点数尺寸, 如

`(10) fixed * [11] cmr12` 当要求的是 10pt 时, 就在 11pt 尺寸下上载 `cmr12`.

上面所有函数都可以前缀一个 `s`(表示无声), 以禁止在屏幕上显示信息。因此 `sub *` 的无声形式为 `ssub *`, 无声的空函数是 `s *`。

现在给出另一个例子, 考虑黑斜打字机字体的定义, 因为在计算机现代字体集中没有这种字体:

```
\DeclareFontShape{OT1}{cmr}{bx}{it}{
  (-) ssub * cmr/m/it }{}
```

该命令会把所有尺寸 `(-)` 的字体(无声地)用中间斜体打字机属性取代。这就是那些由 `\DeclareFontShape` 命令确定的相关属性的字体。

字体定义命令也可以在宏包文件中调用, 甚至在文档中也可以使用。然而, 通常的过程是先把每条 `\DeclareFontEncoding` 命令存贮在一个名为 编码`enc.def` (例如 `OT1enc.def` 对应于 OT1 编码) 的文件中, 然后把命令 `\DeclareFontFamily` 和 `\DeclareFontShape` 放在一个文件中, 其名称由编码加上族标识, 后缀 `.fd` 组成。例如, 编码 OT1 和族 `cmr` 的形状定义可以在 `OT1cmr.fd` 文件中找到。当选择的编码和族组合并没有定义时, `LaTeX` 就会尝试找到相应的 `.fd` 文件做为输入。因此并不需要显式地输入这个文件, 因为必要时可以自动调入。

然而, 事先声明编码是很重要的。如果在当前格式中并不知道编码, 我们就必须调用命令 `\DeclareFontEncoding`, 方法是要么显式地调用, 要么上载 编码`enc.def` 文件。例如, 调用已存在的宏包 `fontenc` 就可以做到这一点:

```
\usepackage[OT2,T1]{fontenc}
```

这里所期望的编码列在中括号内做为选项, 这组选项就是当前的编码。

普通用户从来不必担心这些问题。然而, `LaTeX` 程序开发人员将会发现这些事情是相当容易的。例如, 在 NFSS 中安装 PostScript 字体是相当简单的。一些常用的 PostScript 字体已经在它们自己的 `.fd` 文件中做为单独的族定义好了; 例如 `OT1ptm.fd` 就把 PostScript 的新罗马字体与一个名为 `ptm` 的族关联起来。激活这些字体的宏包包含在名为 `times.sty` 的文件中, 其中主要包含如下几行:

```
\renewcommand{\rmdefault}{ptm}
\renewcommand{\sfdefault}{phv}
\renewcommand{\ttdefault}{pcr}
```

这就使得新罗马 `ptm` 成为默认罗马字体族, 用 `\rmfamily` 命令调用, 而 Helvetica `phv` 成为默认的 sans serif 字体族(用 `\sffamily` 调用), Courier `pcr` 为默认的打字机字体族(用 `\ttfamily` 激活)。

另外两个关于 NFSS 使用的例子是华盛顿大学的 Cyrillic 字体和 Cork 编码中的扩展 DC 字体。在文档中只要选择编码 OT2(对应于 Cyrillic), T1(对应于 DC 字体) 就可以激活这两个字体。(这些编码必须首先进行声明, 例如利用上面说明的 `fontenc` 宏包。)

8.5.9 编码命令

在 \LaTeX 中, 必须用命令来得到特殊的字符和重音, 例如 $\backslash 0$ 显示出 Scandinavian 字母 \emptyset 。这个字符在字体表格中的位置与编码有关 (在 OT1 中是第 31 个字符, 而在 T1 中是第 216 个字符), 因此当改变了编码时, 需要重定义所有这样的符号命令。为此要借助于某种编码命令来完成, 这些命令通常与 $\backslash \text{DeclareFontEncoding}$ 命令一起位于 `编码enc.def` 文件中。

为了定义一个在不同编码下有不同功能的命令, 可以用:

```
 $\backslash \text{ProvideTextCommand}\{\backslash \text{命令}\}\{\text{编码}\}[\text{参数个数}][\text{可省参数}]\{\text{定义}\}$ 
```

```
 $\backslash \text{DeclareTextCommand}\{\backslash \text{命令}\}\{\text{编码}\}[\text{参数个数}][\text{可省参数}]\{\text{定义}\}$ 
```

其行为同 $\backslash \text{providecommand}$ 和 $\backslash \text{newcommand}$ 命令一样, 但只有编码被激活时 $\backslash \text{命令}$ 才具有这里给定的定义。因此对每种编码, $\backslash \text{命令}$ 就具有不同的定义。

```
 $\backslash \text{DeclareTextSymbol}\{\backslash \text{命令}\}\{\text{编码}\}\{\text{位置}\}$ 
```

定义 $\backslash \text{命令}$ 为当编码被激活时字体中给定位置的字符。

```
 $\backslash \text{DeclareTextAccent}\{\backslash \text{命令}\}\{\text{编码}\}\{\text{位置}\}$ 
```

定义 $\backslash \text{命令}$ 为一个重音命令, 当编码被激活时使用位于给定位置处的字符作为重音符号。

```
 $\backslash \text{DeclareTextComposite}\{\backslash \text{命令}\}\{\text{编码}\}\{\text{字母}\}\{\text{位置}\}$ 
```

```
 $\backslash \text{DeclareTextCompositeCommand}\{\backslash \text{命令}\}\{\text{编码}\}\{\text{字母}\}\{\text{定义}\}$ 
```

定义 $\backslash \text{命令}$ 及后接的单个字母的行为是显示在字体给定位置处的字符或者执行定义。在 T1 编码中这些声明是相当有用的, 这时许多重音字母就是单独一个符号。因此 $\backslash \{e\}$ 在字母 e 上放尖重音 (字符 19), 而在 OT1 中它显示的是位置 233 处的字符。这个行为是用如下方式得到的:

```
 $\backslash \text{DeclareTextAccent}\{\backslash '\}\{\text{OT1}\}\{19\}$ 
```

```
 $\backslash \text{DeclareTextComposite}\{\backslash '\}\{\text{T1}\}\{e\}\{233\}$ 
```

相应于给定编码的这条命令必须已经有定义, 方法是调用命令 $\backslash \text{DeclareTextAccent}$ 或者 $\backslash \text{DeclareTextCommand}$ 。对后一种情形, 必须把它定义成具有单个参数值的命令。

上面所有定义命令都生成相应于指定编码中的新命令。如果这里定义的命令在其他编码中调用, 会给出错误信息。可以为未指定的编码给出默认定义:

```
 $\backslash \text{DeclareTextCommandDefault}\{\backslash \text{命令}\}[\text{参数个数}][\text{可省参数}]\{\text{定义}\}$ 
```

```
 $\backslash \text{ProvideTextCommandDefault}\{\backslash \text{命令}\}[\text{参数个数}][\text{可省参数}]\{\text{定义}\}$ 
```

```
 $\backslash \text{DeclareTextAccentDefault}\{\backslash \text{命令}\}\{\text{编码}\}$ 
```

```
 $\backslash \text{DeclareTextSymbolDefault}\{\backslash \text{命令}\}\{\text{编码}\}$ 
```

这里前两条命令创建适用于所有未指定编码的默认定义, 而后两条命令声明把哪种编码取为默认值。

注意: $\backslash \text{DeclareTextCompositeCommand}$, $\backslash \text{Provide} \dots$ 和默认命令是于 1994 年 12 月 1 日加到 $\text{\LaTeX} 2_{\epsilon}$ 中的。因此使用这些命令的文件中应包含:

```
 $\backslash \text{NeedsTeXFormat}\{\text{LaTeX2e}\}[1994/12/01]$ 。
```

8.6 输入编码

在不同的计算机系统上,有可能可以直接向输入文件中输入一些 2.6.5–2.6.7 节的特殊符号,而且在用文本编辑器准备 L^AT_EX 文件时,这些符号会显示在屏幕上。有些 T_EX 实现版本确实可以把这些特殊符号转化成对应的命令,不幸的是它们并没有标准编码,因此这样的文件是与系统有关的。例如,特殊字母 Æ 是 ISO-Latin1 编码方案的第 198 位字符,而在 IBM PC 扩展系统中却是第 146 位字符。

inputenc 宏包可以解决这个与系统有关的问题。其调用方式为

```
\usepackage[编码]{inputenc}
```

这样就依照代码方案 编码 来定义第 128–255 位输入字符的代码。编码 可能取的值为 ascii(没有扩展字符), latin1, latin2 (ISO-Latin1 和 2 代码), cp437, cp850(IBM 代码页中第 437 页和 850 页) 以及 applemac(Apple Macintosh 代码)。

在文件 编码.def 中用如下命令定义了扩展字符:

```
\DeclareInputText{位置}{文本} 或者
\DeclareInputMath{位置}{数学}
```

这样就把 文本 或 数学 的代码指定为给定 位置 的字符。例如, latin1.def 中包含

```
\DeclareInputText{198}{\AE}
```

这样就把输入的第 198 位字符翻译成 L^AT_EX 的 \AE 命令。

这也就是说,为了得到 the rôle of the æther, 可以直接输入 the rôle of the æther (不必用 the r[^]{o}le of the {ae}ther)。但是注意,这里即使同样的输入,在不同的计算机系统上结果可能完全不一样。

8.7 特殊符号的替代

在 CM 和 DC 的字体布局方案中有很多符号,是不能直接用通常的命令输入的。例如, *ı* 和 *ı* 认为是输入文本 !‘ 和 ?‘ 的连写,其他特殊符号类似。有一些符号对应的命令只能用在数学模式中。

利用连写生成符号的一个问题是有些特殊字体可能并不支持这种操作,即使在相应的位置上有这个符号。

任何符号都可以通过 \symbol 命令直接使用,这条命令要求用布局位置做为参数值,但实际生成的字符与字体编码有关。一些 \text... 命令可以用来直接显示所有这些符号,而与当前编码无关。

另外还有:

\textcompwordmark 表示分开连写的不可见字符

\textcircled{字符} 用圆把字符包起来,如 ⊗。

字符 生成当前文本(不是数学)字体中的上标。

命令	符号	替代
连写		
<code>\textemdash</code>	—	---
<code>\textendash</code>	–	--
<code>\textexclamdown</code>	¡	!‘
<code>\textquestiondown</code>	¿	?‘
<code>\textquotedblleft</code>	“	‘‘
<code>\textquotedblright</code>	”	’’
<code>\textquoteleft</code>	‘	‘
<code>\textquoteright</code>	’	’
数学符号		
<code>\textbullet</code>	•	<code>\$_bullet\$</code>
<code>\textperiodcentered</code>	·	<code>\$_cdot\$</code>
杂类		
<code>\textvisiblespace</code>	□	<code>\verb** +</code>

8.8 其他标准文件

基本的 L^AT_EX 格式包含了需要生成通常文档所需要的绝大多数功能。然而还存在许多其他功能，在某些应用中是不可缺少的，可以用 `\usepackage` 命令 (3.1.2 节) 把它们包含进来。由从事不同方面工作的用户编写了上百个这样的宏包，可以通过计算机网络取得这些文件，还有一些宏包是由 L^AT_EX3 项目组的成员开发的，与标准安装版本一起发行；另外有些宏包则是标准安装版本的一部分。

8.8.1 特殊文档

在标准安装版本中有许多特殊的‘文档’，这些文件的后缀为 `.tex`。它们有：

`small.tex` L^AT_EX2.09 中一个简短的示例文档；

`sample.tex` L^AT_EX2.09 中一个较长的示例文档；

`small2e.tex` L^AT_EX 2_ε 中一个简短的示例文档；

`sample2e.tex` L^AT_EX 2_ε 中一个较长的示例文档；

`labl1st.tex` (8.3.1 节) 显示出所有的交叉引用表的翻译及对应页码；它会交互式地询问要列出的文档名称；

`idx.tex` (8.3.3 节) 显示出文档中所有的 `\index` 项，以两列方式同时显示出页码；它也要交互式地询问要列出的文档名称；

`testpage.tex` 测试文本在页面上的位置；这也是一个打印机驱动器的检测程序，以确保正确的页边界，恰当的放缩显示；在 L^AT_EX2.09 中，这个文档只适用于美国信纸，而在 L^AT_EX 2_ε 中，它会交互式地询问所希望的纸张尺寸；

`nfssfont.tex` 这个文档显示字体表格，示例文本以及其他各种检测；它会交互式询问字

体的名称; `\help` 命令会显示出所有可用的命令;
`docstrip.tex` 如其说这是一个文档, 不如说它是一个程序; 它是解开文档源文件, 从而得到操作文件的基本工具; 它不但去掉注释, 而且会根据不同的选项加上替代编码。

8.8.2 其他类

除了我们已经讨论的标准类 (`book`, `report`, `article`, `letter`) 外, 在标准安装版本还有如下类:

`proc` 生成会议学报的照像制版拷贝; 它是 `article` 类的两列模式的变体; 它包含额外的一条命令 `\copyrightspace`, 用来在第一列的底部留下显示版权信息的空间; (在 $\text{\LaTeX} 2.09$ 中, 这是 `article` 样式的一个选项, 在 $\text{\LaTeX} 2_{\epsilon}$ 中也可以这样用, 但这只是为了兼容的需要);

`slides` (8.9 节) 相应于 $\text{\LaTeX} 2.09$ 中的 `SL\TeX`, 用来生成演讲材料;

`cctart`, `cctbook` CCT 中文 $\text{\LaTeX} 2_{\epsilon}$ 所提供的汉化 `article` 和 `book` 类。详情见第十章。

8.8.3 标准宏包

那些成为 \LaTeX 标准安装版本一部分的宏包, 考虑得相当广泛, 这样我们就可以期望总是可以用它。而其他的宏包, 即使是 $\text{\LaTeX} 3$ 项目组的, 也都只是供选择应用的。

有些标准宏包我们在其他地方已做了描述。标准宏包有:

`alltt` 给出了 `alltt` 环境 (4.8 节), 它非常类似于 `verbatim` 环境, 只是 `\{` 和 `\}` 的行为与通常的一样; 即在文本中的命令要被执行, 而不是照原样显示;

`bezier` (6.4.9 节) 提供了在 `picture` 环境中画二次曲线的能力; 这个功能现在内植于 $\text{\LaTeX} 2_{\epsilon}$ 中, 只是为了兼容原来的文档, 而提供了一个空文件;

`doc` 提供了建立 \LaTeX 宏包文档的特殊功能;

`exscale` 允许以不同尺寸显示的数学符号也根据在 `\documentclass` 命令中的字体尺寸选项进行放缩; 通常这些符号的尺寸是固定的, 与文本的基本尺寸无关;

`fontenc` (8.5.8 节, 第 200 页) 通过上载 `.def` 文件来声明字体编码; 编码的名称列在选项中;

`graphpap` (6.5.6 节) 给出了命令 `\graphpaper`, 以生成 `picture` 环境中的坐标网格;

`ifthen` (7.3.5 节) 允许文本或命令的定义与各种条件的状态有关; 我们可以测试布尔值开关的状态, 计数器或长度的值, 或者某一命令的定义文本;

`inputenc` (8.6 节) 允许在制作 \LaTeX 文档时特殊字符的输入采用的是与系统无关的编码方案;

`latexsym` 用 `lasy` 字体上载 11 个特殊的 \LaTeX 符号, 这些符号现在不再内植于 $\text{\LaTeX} 2_{\epsilon}$ 中; 在兼容模式中这个文件是自动读入的;

`makeidx` (8.4 节) 给出了当用 `MakeIndex` 程序生成关键词索引时可能用到的命令;

`newfont` 重定义了两字母声明, 使得它们的行为与第一版 NFSS 中的一样; 这些字体声明只改变一个属性; 即 `\bf` 类似于 `\bfseries`;

`oldlfont` 定义两字母 (如 `\bf`) 声明的行为同在 \LaTeX 2.09 中文本和数学模式中一样; 这些声明现在并不内植于 \LaTeX 2 ϵ 中, 但所有的标准类中都有其他义; 保留这个宏包只是处于兼容性的考虑;

`pict2e` (6.5.6 节) 通过去掉直线长度、粗细、斜角以及圆的半径等方面的限制, 来增强某些 `picture` 环境命令的功能, 它利用了与驱动器有关的功能, 但现在还没有实现;

`showidx` (8.3.3 节) 使得 `\index` 项显示为边注;

`shortverb` 提供了 `\MakeShortVerb` 和 `\DeleteShortVerb`, 这两条命令用来设定和删掉某一特定字符的功能与 `\verb` 功能等价, 用以显示源文本。这个字符前缀反斜杠做为命令的参数。因此如果进行了如下调用:

```
\MakeShortVerb{\|}
```

那么 `| \cmd{ } |` 等价于 `\verb| \cmd{ } |`, 结果都是 `\cmd{ }`;

`syntonly` 提供了命令 `\syntaxonly`, 在导言中调用它, 不把结果输入到 `.dvi` 文件中, 但会给出错误和警告信息; 这样会比正常的情形快四倍; 在并不急于得到输出时, 这种方法可以用来检验处理过程;

`Tienc` 把 DC 字体编码设为标准;

`tracefmt` 是一个检验 NFSS 字体的对话式工具; 在 `\usepackage` 命令中可以用下面这些选项控制其行为:

`errorshow` 不在屏幕上显示所有的警告和提示信息, 但会送到抄本文件中; 在屏幕上只显示错误信息;

`warningshow` 在屏幕上显示警告和错误信息; 这种形式与完全没有用这个宏包很相像;

`infoshow` (默认选项) 在显示器上显示字体选择信息, 通常这些信息只是写到抄本文件中;

`debugshow` 在每次改变字体后, 都会向抄本文件中写入相当多的信息; 这样会得到一个很大的抄本文件;

`pausing` 把所有的警告信息都转化为错误信息, 这样会使得处理暂时中止, 等待用户的反馈;

`loading` 显示对外部字体的上载。

8.8.4 其他有用的宏包

\LaTeX 3 项目组的成员个人也编写了许多宏包, 为了便于使用, 这些宏包收集在 CTAN 上一个名为 `tools` 的特殊目录中。其中绝大多数是从 \LaTeX 2.09 时代就开始创办的, 现在已更新到了 \LaTeX 2 ϵ 。

我们这里扼要介绍一下其中某些宏包能做什么事; 要想详细了解它们的功能, 只要用 \LaTeX 处理其 `.dtx` 文件就可以了。

`array` 是对标准 \LaTeX 中 `tabular` 和 `array` 环境 (6.6.1 节) 的重新实现, 增加了许多功能。除了等价于 `\parbox[t]{宽度}` 的列定义 `p{宽度}`, 还有 `m{宽度}` (类似于 `\parbox{宽度}`) 和 `b{宽度}` (类似于 `\parbox[b]{宽度}`), 这三个定义都用给定的

宽度 做为列的宽度, 只是各列相对于顶行, 中间和底行对齐。也可以用 `>{声明}` 和 `<{声明}` 在一列的开始和结尾处插入声明, 可以用这种方式来包含一个字体声明, 作用于列中所有行, 或者在数学模式中进出。也可以定义用户自己的列类型。

`dcolumn` 需要 `array` 宏包, 在 `tabular` 表格中可以用小数点对齐。

`delarray` 需要 `array` 宏包, 能在 `array` 环境外面加上大的括号符号, 这样可以生成各种形式的矩阵。

`hhline` 需要 `array` 宏包, 在表格的水平直线输入方面更富有弹性。

`longtable` (不需要 `array` 宏包, 但认识其功能) 使我们可以创建长达几页的表格, 中间自动进行分页。

`tabularx` 定义了 `tabularx` 环境, 其类似于 `tabular*`, 生成具有希望宽度的表格, 但只是伸展列宽, 而不是伸展列间距。

`afterpage` 允许存贮代码, 并插入在当前页的尾部。

`enumerate` 重新实现了 `enumerate` 环境, 可以用一个可省参数值确定每个 `\item` 生成数字的样式。

`fileerr.dtx` 解开这个文件, 会生成一组小文件, 文件名称为 `h.tex`, `e.tex`, `s.tex` 和 `x.tex`, 可以用来解决 \TeX 中文件不存在时的情形。例如, 回答 `x`(回车) 会上载最后那个文件, 从而结束 \TeX 坚持要求输入文件名的状态。

`fontsmpl` 是一个打印希望了解的字体示例的宏包。

`ftnright` 在两列模式中把脚注放在列的右端。

`indentfirst` 对章节的第一段进行缩进, 通常 \LaTeX 并不进行这种操作。

`layout` 定义了命令 `\layout`, 它生成当前页面格式的示意图, 显示出各个参数的值。

`multicol` 给出了 `multicol` 环境, 它是 `\twocolumn` 命令的扩充, 把其文本以指定数目的列显示出来, 各列长度均衡, 而且在开始和结尾部分不开始新页。

`showkeys` 显示出由 `\label` 和 `\bibitem` 定义、并由 `\ref`, `\pageref` 和 `\cite` 使用的所有交叉索引关键词; 它们以边注和行间注释的形式出现, 只适用于草稿, 以检查交叉索引的情况。

`somedefs` 根据 `\usepackage` 中选项, 在宏中只允许定义选定的命令。

`theorem` 推广了 `\newtheorem` 命令, 使之成为更富有弹性的“定理型”环境。

`varioref` 定义了 `\vref` 和 `\vpageref` 命令, 分别类似于 `\ref` 和 `\pageref` 命令, 它们会检测被引用对象是否就在前一页, 如果是这样的话, 就显示出类似于“on the previous page”这样的文本。

`verbatim` 重新实现了 `verbatim` 环境, 避免了长文本时的内存溢出; 而且还定义了命令 `\verbatiminput`, 以输入一个文件, 并且按字面显示文本, 还有一个 `comment` 环境, 以生成源文件中的注释块。

`xr` 利用这个宏包可以用 `\ref` 交叉索引另外文档中的 `\label` 命令。

`xspace` 这个工具可以更正命令名称吞掉后接空格的问题; 因此如果命令 `\PS` 的定义为 `\newcommand{\PS}{PostScript\xspace}`

那么 `\PS file` 用法就完全可以, 没有必要用 `_` 或 `{ }` 结束 `\PS` 命令。

要注意一点，如果是在 DOS 操作系统下面（如 CCT emTeX）使用上述宏包，那么由于文件名的限制，需要对某些文件名太长的宏包（如 `enumerate.sty`）重命名。这不可避免地降低了 L^AT_EX 文档的普适性。关于重命名文件的操作方法，见相关操作系统手册。

8.8.5 其他软件

在 L^AT_EX 的网络服务器上有很多平行的目录，其中包含相当多的独立于标准安装版本之外的附属软件。这些软件都属于半正式状态。

amslatex 是一组由美国数学会提供的用于高级数学排版的宏包。我们在第五章中部分介绍了其数学排版功能。另外，它还定义了许多自己的类，向有些杂志投稿时，可能会需要这些类。

babel 提供了 L^AT_EX 中的多语种支持。

graphics 使得可以利用一些宏包，来进行图形处理：插入、旋转和处理外部图形和图示，以及显示彩色。在第 8.11 节对其进行了介绍。

mfnfss 提供在 NFSS 中安装其他 METAFONT(位图) 字体的宏包。

psnfss 提供在 NFSS 中安装特定 PostScript 类型 1 字体的宏包。

tool 是一组由 L^AT_EX3 项目组成员编写的实用宏包。

上面宏包中大多数都随软件附有完整的文档。

并不是只有 L^AT_EX3 项目组的成员在开发 L^AT_EX 类和宏包文件。在网络服务器上有成千上万个已公开的个体宏包。其中大多数很有用的宏包在 [4] 和 [5] 两书中有介绍。

可以编写自己的类和宏包的能力 (附录 C) 是 L^AT_EX 的巨大能量之一。与其期望初始发明者把文本处理中每件事都做得很好，不如实际用户针对自己专业需要，寻找方法，并使得其他用户也可以应用自己的这一方法。

8.8.6 L^AT_EX 网络服务器

现在，有许多网络服务器提供 L^AT_EX 宏包，或者与 T_EX/L^AT_EX 有关的程序。当然，这些宏包和软件都是免费的。最常用最及时的几个地址为：

所在国家	IP 地址
美国	<code>pip.shsu.edu</code>
英国	<code>ftp.tex.ac.uk</code>
德国	<code>ftp.dante.de</code>

从这些站点上可以取得现行的 L^AT_EX 实现程序 (例如最新版本的 emTeX)，以及 L^AT_EX 宏包以及其他一些辅助程序。另外有些接受 L^AT_EX 格式稿件的杂志或期刊，也把自己的类文件或宏包放在这些站点上。上述三个站点的内容是一样的。

另外，如果我们想搜索某个宏包 (例如 `alltt.sty`，第 204 页)，那么可以用 IE 或 NetScape 浏览

<http://www.tex.ac.uk/CTANfind.html>,

在 Search CTAN for a File 下面的编辑框中，输入所需要的宏包文件全名，这里要输入的就是 `alltt.sty`。点按提交钮。在新出现的窗口中，选择一个 CTAN 地址，可以是上

面列出中的一个, 或者其他的离你最近的地址。这样, 它就会告诉你如何得到该宏包。这里的 CTAN 代表 Comprehensive T_EX Archive Network。

在国内, 目前还没有 CTAN 的镜像站点。作者目前在中国科技大学数学系的科学计算与计算机图形学实验室主办了一个名为 T_EXGuru(<http://202.38.68.78/~texguru/>) 的站点, 这是国内少有的专门以 L^AT_EX 为主题的主页。如果你有问题, 可以访问该站点。实际上本书有些内容就是对站点内容的总结和提炼。

另外, 在 <ftp://ftp.cc.ac.cn/pub/cct/> 上可以下载最新的 CCT 中文 L^AT_EX(第十章)。

8.9 报告材料的准备

在 L^AT_EX2.09 中存在另一种 T_EX 版本, 它与 L^AT_EX 是平行的, 但名称为 S_LT_EX, 可以用它生成报告材料, 如制作彩色的幻灯片或视图。由于这个特殊的 L^AT_EX 变体利用了不同的字体集, 从而生成的结果比通常的书籍样式生成的结果更适合于进行投影。然而, 这需要另一种 T_EX 格式(1.1.1 节), 因为原来版本的 L^AT_EX 把某个字体集显式地程序设计在格式文件中, 所以用 S_LT_EX 以示与通常 L^AT_EX 的区别。运行的是 S_LT_EX, 而不是 L^AT_EX, 实际上差别就在于前者用的是 `splain` 格式来调用 T_EX 程序, 而后者则用的是 `lplain` 格式。

在 L^AT_EX 2_ε 中利用新字体选择方案, 我们可以在一个 L^AT_EX 格式中替换所有的字体, 而不会有任何问题。因此现在不再需要让 S_LT_EX 成为一个独立于 L^AT_EX 之外的事物。为了生成报告材料, 我们只需要在 L^AT_EX 2_ε 中使用 `slides` 类就可以了。在 L^AT_EX 2_ε 中该类同其他类的选择方式完全一样, 而在 L^AT_EX2.09 中, `slides` 样式只能在 S_LT_EX 中选用, 而且事实上它也只接受这一种样式。

从此当我们用 S_LT_EX 这一名称时, 它只是相对于 L^AT_EX2.09 而言的, 而它与 L^AT_EX 2_ε 中的 `slides` 类有相当大的差别。

8.9.1 slides 类

我们所谓的报告材料, 就是那些要投影给观众看的幻灯片。我们当然可以利用通常的 L^AT_EX 命令编写这些材料, 但是在如何得到适当的字体尺寸, 合理安排覆盖, 确保文本不会出现不期望的分页, 以及书籍样式的字体是否适合于投影等方面存在一些问题。L^AT_EX 类 `slides` 就是尝试解决这些问题的。

在这个类中用的是一组不同的字体集, 其中的小写字母比通常的字体要大一些, 而且基本尺寸更大。这当然会限制了一页 / 片上的文本量, 但是这对做报告材料就非常好。这里的文本应局限于关键词和简略的句子。满满的一页普通文本即使投影在屏幕上, 也不会有观众去看。

投影片也应该利用彩色。从这一点上来看, S_LT_EX 就更应该更新了。当在 80 年代中期人们创造出第一个 L^AT_EX 版本时, 还没有理想价格的彩色打印机可以使用。一种变通的方法是利用彩色层, 也就是每种颜色的文本(用黑白色)单独打印出来, 然后把它复

印到透明片上,接着把它与彩色纸叠加起来就可以了。到了今天,有更好的可以直接生成彩色投影片的方法。虽然对彩色的支持并不是 $\text{\LaTeX} 2_{\epsilon}$ 核心的一部分,但是宏包 `color` 却可以针对某些驱动器提供彩色命令。在 $\text{\LaTeX} 2_{\epsilon}$ 中任何地方都可以用这个宏包,并不是只能用在 `slides` 类中的。

即使没有彩色的功能, `slides` 在利用特殊字体和其他长处于黑白视图生成等方面也是非常有用的。通常 \LaTeX 的格式命令大多都可以使用,只有分页和章节命令例外。在下面一节中描述只属于 `slides` 的特殊功能。

在 $\text{\LaTeX} 2_{\epsilon}$ 中并没有提供彩色层命令,只有兼容模式才可以用它。

8.9.2 制作幻灯片的环境

生成幻灯片的源文件在结构上同通常文档类似,只是这里用的是 `slides` 类:

```
\documentclass{slides}
  导言文本
\begin{document}
  幻灯片文本
\end{document}
```

如果希望得到真正的彩色(不是黑白加彩色膜),那么在导言中可以按如下方法加入 `color` 宏包:

```
\usepackage[dvips]{color}
```

在导言中可以包含某些全局性规定,如像通常那样改变纸张尺寸或选择纸张样式。切记在导言中不能包含任何可显示的材料。

所有出现在 `\begin{document}` 命令之后,但在下面将要描述的特殊幻灯片环境之前的文本,都输入到一个没有编号的首页上,它位于所有幻灯片的前面。可以用它做为幻灯片的封面。

在组织幻灯片不同部分时,有三个环境可以使用:主幻灯片自身,可能用的覆盖,和对幻灯片的注解。

幻灯片

幻灯片或视图是用如下环境生成的:

```
\begin{slide} 文本和命令 \end{slide}
```

`slide` 环境中的内容可以是任意所期望的文本。注意 `slides` 类使用的是自己专用的字符字体集,正常尺寸中的标准字体大致相当于 \LaTeX 的 `\LARGE` 尺寸的 sans serif 字体 `\sf family`。在这个环境中可以使用 4.1 节中的通常字体命令和声明,以及第四章中的其他显示和列表环境。然而,在环境中不能出现分页,因为整个文本希望在一页(幻灯片或视图)上结束。后面的 `slide` 环境会顺序编号的。如果一页出现了溢出,会显示出警告信息。

可以使用来自于 `color` 环境的彩色命令,前提条件是这个宏包已经被调入,这样就可以直接输出彩色,而不必采用以往 \LaTeX 中的彩色膜。

覆盖

所谓覆盖就是幻灯片的补充，它可以放在幻灯片上面，以弥补某些不足。基本想法就是在报告时通过稍后才填上某些关键词，或者能够替换某些文本，来创造出某种悬念。

`slides` 类是利用 `overlay` 环境生成覆盖的，其作用方式同 `slide` 环境完全一样，只是编号是前面幻灯片的子编号。因此跟在第 6 号幻灯片后面的覆盖编号为 6-a, 6-b 等等。

`\begin{overlay}` 文本和命令 `\end{overlay}`

生成视图的 `slide` 环境若要有覆盖，那它应在相应的 `overlay` 环境之前。幻灯片和覆盖环境中应包含相同的文本，只是某些部分用下面的声明以不可见的方式显示：

`\invisible` 和 `\visible`

这两条命令的作用方式与字体声明一样。可以把它们放在大括号 `{}` 内以限制其作用范围，也可以用于整个环境。通常在 `slide` 环境中只会使某几个单词不可见，而在相应的 `overlay` 环境中，在开头处放上 `\invisible` 声明，则只有幻灯片中看不见的部分才是可见的。

覆盖的一种应用就是提供替换文本。例如我们可以显示一张成本估计的表格，然后把图示放在覆盖上。通过交换这些覆盖，只要进行某些程序修改，所得到的新数据就会适合于同样的表格。

注解

在进行实际的幻灯片投影时，经常需要解释某些关键词或者添加其他注解。`slides` 类提供了 `note` 环境，可以生成给报告人的提示。

`\begin{note}` 文本 `\end{note}`

同 `overlay` 一样，注解也是相对于前面的幻灯片进行子编号的，只是这里用的是数字，不是字母。例如，接在第 4 号幻灯片后面，注解的编号为 4-1, 4-2, 等等。

8.9.3 其他功能

页面样式

\LaTeX 命令 `\pagestyle{样式}` 也可以用在 `slides` 类中。可以使用下面这些样式：

plain 所有的幻灯片、覆盖和注解的编号都显示在右下角。

headings

同 **plain** 一样，只是如果选择了 **clock** 选项（见下），就会在注解的左下角显示一个时间标记。这也是默认的页面样式。

（**headings** 和 **plain** 样式在 $\text{SL}\text{\TeX}$ 中有很大的差别，前者这时还会绘制对齐标志。）

empty 打印出来的页面上没有页码（或者对齐标志）。

在通常的 \LaTeX 中，`\pagestyle` 命令是在导言中调用的，从而对整个文档都有效。单个页面可以利用 `\thispagestyle` 命令给出不同的样式，这条命令只对当前一页有效。在上面环境内部不能用这条命令，而 `\pagestyle` 命令却可以用在 `slide`, `overlay` 和 `note` 环境外面。

时间注解

在 `\documentclass` 的选项中可以选择 `clock`, 它激活如下两条命令:

```
\settime{秒数}
```

```
\addtime{秒数}
```

这会以分钟为单位在注解的底部显示出时间。通过这种方法, 可以提醒报告人在该处可以做怎样程度的讲述。时间标志的值是用上面那两条命令进行设置和增加的, 这里的参数值以秒为单位。记时器的初始值为 0。

有选择地处理幻灯片

如果幻灯片文件处理的 `slide` 环境很多, 而用户可能只想修改其中很少一部分的幻灯片, 这样就没有必要再全部重新输出一遍。为此, 可以用下面的命令做到这一点:

```
\onlyslides{幻灯片清单}
```

把这条命令放在导言中。这里的 `幻灯片清单` 表示升序的一组幻灯片编号, 例如, 2, 5, 9-12, 15, 用来指定要处理的幻灯片或幻灯片范围。

不存在的幻灯片编号也可以出现在 `幻灯片清单` 中。比如说在幻灯片文件中只有 20 张幻灯片, 那么 `\onlyslides{1,18-999}` 命令将会使得只有第 1 号和 18-20 号幻灯片被处理。属于这些幻灯片的覆盖也同时被输出。

最后, 在导言中的命令

```
\onlynotes{注解清单}
```

使得只有列在 `注解清单` 中的注解才会被输出。假设第 5 号幻灯片有三个与之相关联的 `note` 环境, 那么 `\onlynotes{5}` 将会使得只有页码为 5-1, 5-2 和 5-3 的注解被输出。

`\onlyslides` 和 `\onlynotes` 命令与在 8.1.2 节中描述的 `\includeonly` 命令在功能上很相似, 也可以与 8.1.3 节中描述的通过利用 `\typein` 命令一样, 实现幻灯片和注解的交互式选择:

```
\typein[\slides]{Which slides to do?}
```

```
\onlyslides{\slides}
```

在处理过程中会在屏幕上显示出 `'Which slides to do?'` 的信息, 这时用户就可以输入所期望处理的幻灯片。对 `\onlynotes` 也可以类似操作。

8.10 书信的编辑

除了前面讲到的几种标准 L^AT_EX 文档类外, 还有另外一种文档类, 名为 `letter.cls`, 专门用来编辑书信。但是 `letter.cls` 只限于书写没有任何装饰 (例如信笺头或者公司名称) 的私人信件。

单个输入文件中可以包含不只一封信和地址的文本, 这些信都是来自于同一个寄信人。如果愿意的话, 可以同时自动打印出地址标签。绝大多数普通的 L^AT_EX 命令在 `letter` 类中的功能仍像通常一样。然而一个例外就是章节命令, 在这里它会导致错误消息: !

Undefined control sequence. 实际上在书信中也没多大必要来划分章节。另一方面, 也有许多只适用于这一样式的特殊命令。

一封信的源文件开头与所有 L^AT_EX 文档类似, 即

```
\documentclass[选项]{letter}
```

这里的选项可以取列在 3.1 节中的所有种类, 只是 twocolumn 和 titlepage 例外, 因为在书信中没有必要用到它们。在 L^AT_EX 2_ε 中也可以用 twoside 选项, 但在 L^AT_EX 2.09 中不能用这一选项。

每封信都必须有寄信人的姓名和地址, 这两类信息是通过如下放在导言中的命令来给出, 从而适用于同文件中的所有信件:

```
\address{ 寄信人地址 }
```

```
\signature{ 寄信人姓名 } 或者 \name{ 寄信人姓名 }
```

寄信人地址通常由几行组成, 行与行之间用 \\ 分开, 例如:

```
\address{Scientific Computing and Computer Graphics\\
          Department of Mathematics\\
          University of Science and Technology of China\\
          Hefei, Anhui, 230026\\China}
```

如果给出了 \name 命令中的条目, 那它将被用做信头的回信地址。而放在 \signature 命令中的条目将显示在书信结尾处给作者签名用的空白的下面。如果没有给出 \signature, 就会在这里插入 \name 中的内容。这样就可以得到一种非常正式的回信地址以及另一种不同形式的 (可能是多行的) 签名:

```
\name{Dr. J.\ S.\ Deng}
\signature{JianSong Deng\\Lab Manager}
```

如果在导言中调用了上述命令, 那么它们就会对文档中的所有信件都适用, 除非某些信件中以新内容调用了这些命令。因此有的信件可以拥有与其他信件不同的签名。这些条目的适用范围只延续到调用它环境的结束 (见 7.5.4 节)。

在标准 L^AT_EX 的 letter 类中还可以包含另外两个寄信人条目。可以利用它们进行适用于局部范围的修改。基本想法就是如果没有调用 \address, 就生成已设计好的公司信笺头以及寄信人的房间号和 / 或电话号码。因此提供了命令

```
\location{房间号} 和 \telephone{电话号码}
```

当使用的是 letter.cls, 而且没有调用 \address 时就会在每页的底部显示出房间号和电话号码。

导言中也可以包含 \pagestyle 命令, 选项与通常的一样, 即 plain, empty 或 headings. 第一个为缺省值, 它在第一页后面所有页的底部居中加上页码。headings 页面样式会在第一页后面所有页的顶部在一行中加上收信人姓名, 日期以及页码。

在导言命令后面, 就是同所有 L^AT_EX 文件一样, 用 \begin{document} 命令开始实际的正文。正文由一封或多封信件组成, 每封信用如下语法包围在 letter 环境中:

```
\begin{letter}{收信人信息} 信件内容 \end{letter}
```

其中收信人信息由收信人的姓名和地址组成, 可以用 \\ 命令分成几行。

```
\begin{letter}{Mr. Donald J. Burns\\
               Ontario Institute of Physics\\
               41 Adelaide St.\\
               London, Ontario\\Canada N4R 3X5}

\end{letter}
```

信件内容通常以 `\opening` 命令开头, 以 `\closing` 命令结尾, 这两条命令之间就是信的主体, 可以包含所有要用的 \LaTeX 命令。这两条命令的语法是

```
\opening{ 问候 }
```

```
\closing{ 祝好 }
```

其中 问候 是信件的开头称呼, 例如 *Dear Mr. Smith*, 而 祝好 表示正文的结束, 例如 *Yours sincerely*.. `\opening` 命令中也可以包含其他的文本, 例如, 可以用它生成一个主题行, 而真正的问候语是放在接下来的正文中。

\LaTeX 把寄信人的姓名和地址放在第一页的右上角, 其下面是右对齐的当前日期。然后把收信人的姓名和地址靠左边摆放, 后接问候语与信件正文。整个信件以结束语来完成, 然后是寄信人的姓名或签名, 它们相对于中心线左对齐, 并与正文间留下足够的竖直间距, 以供手写签名。

在 `\closing` 命令后, 还有一些命令可以使用, 它们也是信件的一部分。其中一条命令是 `\cc`, 它生成该信件的分发名单:

```
\cc{ 姓名一 \\ 姓名二 \\ ... }
```

文本 ‘cc:’(或者更准确地说, 是在 `\ccname` 中定义的文本) 显示在左页边, 然后向里缩进一点, 并显示信件分发名单。

另一条命令是 `\encl`, 它给出随信附件清单:

```
\encl{ 附件一 \\ 附件二 \\ ... }
```

单词 ‘encl:’(在 `\enclname` 中的文本) 显示在左页边, 随后接上附件的清单。

最后, 命令 `\ps` 可以在签名后加上一条附言。这条命令自身并不生成任何文本, 也没有任何参数。附言文本可以是介于 `\ps` 和 `\end{letter}` 命令之间的任何内容。

通常信件是自动标上当前日期的。然而, 如果希望信件的日期向后推迟一下, 或者把日期固定下来, 那么可以用命令

```
\date{日期文本}
```

其中 日期文本 就放在日期应该位于的地方。

一个书信文件中可以包含任意数目的 `letter` 环境, 每个环境对应于一封信。正如前面已讲到的, 当在导言中调用了 `\address`, `\name` 和 `\signature` 命令时, 它们就会对整个文件中所有信件起作用。当然也可以在一个 `letter` 环境内部 `\opening` 命令前面重新调用上述命令, 以改变寄信人的信息, 这些修改就只对这封信有效。如果已经声明了 `\name` 和 `\signature`, 那么后者就会显示在签字空白的下面。

信件的首页总是没有页码的, 后面的几页才有一个在底部的居中页码(缺省值)或者在顶部有一个包含收信人姓名、日期以及页码的页眉(页面样式 `headings`)。在导言中利用命令 `\makelabels`, 用户可以打印出地址标签。其上的地址信息由 `letter` 环境中的收

信人姓名和地址参数值组成。标准的 letter 类设计的标签尺寸为 $4\frac{1}{2} \times 2$ 英寸，并以两列形式排列。在其他格式中可以改变这一点。要想打印出一张没有对应信件的标签，可以采用如下形式的空白 letter 环境：

```
\begin{letter}{收信人}\end{letter}
```

8.11 插入外部图形

L^AT_EX 提供的绘图功能并不是很强。计算机发展到今天，已经有很多软件，可以帮助我们绘制相当复杂的插图，因此我们有必要让 L^AT_EX 利用这些图形。另一方面，我们有时还会想把一幅照片插入在 L^AT_EX 文档中。但是 L^AT_EX 并没有直接提供包含外部图形的命令，我们需要利用 T_EX 命令 `\special` 来实现上述功能。而 `\special` 的使用，需要对 DVI 文件的格式具有相当的了解，因此我们可以借助于一些高级宏包达到上述目标。

在此处我们假定所讨论的图形都已经通过某种途径，转化为 PostScript 格式。对每种图形软件，这种转化步骤可能不同，必要时请参考软件的使用手册。另外，关于 PostScript 文件的格式，请见 [13]。

第一步，利用 L^AT_EX 宏包在正文中嵌入图形。这时有如下几种可选途径：

1. 利用 `graphics` 宏包嵌入图形，这时可用的命令为

```
\includegraphics[llx, lly][urx, ury]{文件名}    或者
\includegraphics*[llx, lly][urx, ury]{文件名}
```

这里的 `[llx, lly]` 和 `[urx, ury]` 分别指的是图像左下角和右上角的坐标，其默认单位为 pt， $1\text{ in} = 72\text{ pt}$ ，即 `[1in, 1in]` 与 `[72, 72]` 等价。如果只给出了一组位于方括号中的值，就认为给出的是右上角的坐标，而假定左下角位于 `[0,0]`。如果两组坐标都没有给出，那么就从外部 PS 文件中获得相关信息。

有星号形式的命令，会把所有位于指定范围外部的图像部分剪切掉。

注意：要使用上述两条命令，必须在导言部分插入 `\usepackage{graphics}` 命令。

2. 如果用的宏包是 `graphicx`，那么 `\includegraphics` 命令的语法有了很大增强，具体形式为

```
\includegraphics[关键值列表]{文件}    或者
\includegraphics*[关键值列表]{文件}
```

这里的星号形式命令是出于兼容性而提供的，因此它等价于 `clip` 关键值。

关键值列表是由一组用逗号分开的形式为 `关键词=值` 的字符串组成。对于布尔值，如果只指定了关键词，那就等价于 `关键词=true`，没有指定关键词，就等价于 `关键词=false`。下面是关键词类型列表：

- `bb` 图像的“包围盒”，其值必须是四个尺寸，用空格分开。
- `bbllx` 左下角的 x 坐标，只是为了兼容才保留该关键词。
- `bbllly` 左下角的 y 坐标，只是为了兼容才保留该关键词。
- `bburx` 右上角的 x 坐标，只是为了兼容才保留该关键词。

- bbury** 右上角的 y 坐标, 只是为了兼容才保留该关键词。
- hiresbb** 这个关键词让 L^AT_EX 在 PS 文件中搜索 `%%HiResBoundingBox` 注释, 而不搜索 `%%BoundingBox`. 有些程序用这种方法指定更精确的包围盒, 因为这些数通常只取整值。该关键词是一个布尔值, 取 `true` 或 `false`.
- viewport** 这个关键词有四个参数值, 但此时原点的定位是相对于指定文件中的包围盒。因此为了查看图形左下角大小为 20 pt 的方形, 可以用 `viewport=0 0 20 20`.
- trim** 与 `viewport` 类似, 但四个尺寸对应于插入图形在左、底、右、顶四边上被去掉的长度。
- natheight** 图形的自然高度。
- natwidth** 图形的自然宽度。可以通过 `natheight` 和 `natwidth` 把图形的左下角放在 (0,0) 处, 右上角为 (`natheight`, `natwidth`), 方法就是把 `bb=0 0 w h` 中的 `w` 和 `h` 取成与 `natheight` 和 `natwidth` 相同的值。
- angle** 旋转角, 以度为单位, 反时针为正向。
- origin** 进行旋转时的转点。
- height** 需要的高度, 插入图形要放缩到这个尺寸。
- width** 需要的宽度, 插入图形要放缩到这个尺寸。
- totalheight** 需要的总高度, 插入图形要放缩到这个尺寸。如果图形旋转超过 90°, 就不要用 `height`, 而使用这个关键词, 因为这时高度变成了 `depth`。
- keepaspectratio** 布尔变量。当取值为真时, 要求在进行放缩时, 不会发生变形。
- scale** 放缩因子。
- clip** 相对于包围盒剪切图形。这是一个布尔值。
- draft** 局部切换到草稿模式, 即不显示图形, 只是留下空白。它是一个布尔值。
- type** 图形的类型。
- ext** 包含图像数据的文件的扩展名。
- read** 由 L^AT_EX 读入的文件的扩展名。
- command** 要应用到文件上的命令。

注意, 关键词是按从左到右的顺序读入的, 因此 `[angle=90, totalheight=2cm]` 意味着先旋转 90 度, 然后放缩高度到 2cm; 而 `[totalheight=2cm, angle=90]` 的结果是图形最终宽度为 2cm。

3. 利用 `epsfig` 宏包 (其也包含在 `graphics` 宏包组中) 嵌入图形, 这时可用的命令为

```
\epsfig{file=..., height=..., width=..., clip=, angle=..., %
        silent=, bbllx=..., bblly=..., bburx=..., bbury=...}
```

其中 `..` 表示相应关键词的取值, 其意义为

- file** 插入图形文件的名称, 应为 EPS 格式。
- height** 设定图形的高度, 加上 T_EX 识别的单位。如果没有指定该参数, 那么就取图形的自然高度, 即在 PS 文件中 `BoundingBox` 所在行上的尺寸。如果只指定了宽度, 而没有指定高度, 那么就会按与宽度同样的比例放缩自然高

度。

- width** 设定图形的宽度，加上 \TeX 识别的单位。如果没有指定该参数，那么就取图形的自然宽度，即在 PS 文件中 `BoundingBox` 所在行上的尺寸。如果只指定了高度，而没有指定宽度，那么就会按与高度同样的比例放缩自然宽度。
- bbllx** 包围盒左下角的 x 坐标。
- bbllly** 包围盒左下角的 y 坐标。
- bburx** 包围盒右上角的 x 坐标。
- bbury** 包围盒右上角的 y 坐标。
- clip** 这个参数确保不会有部分图形超出包围盒的边界。“clip=” 只是一个切换开关，不取任何值，但不能省略 = 号。
- angle** 确定旋转角，以度为单位，反时针方向为正向。
- silent** 使 `\epsfig` 命令“无声地”工作。

上述三种方法都可以很方便地插入外部图形，而且我们也可以把它们放在 `figure` 环境中，以构造浮动插图。关于浮动对象的操作，请见 6.7 节的讲述。

当我们用上述三种方法中的一种插入了图形后，就可以进行第二步的操作。首先像通常文档一样，用 \LaTeX 对其进行编译，得到 DVI 文件。但是目前在微机上还没有 DVI 浏览器可以直接查看内部嵌有外部图形的输出。因此，为了查看其中的图形，必须把 DVI 文件转化为 PS 文件。我们在 9.4 和 10.5 节中讲解如何进行这一转化。

第三部分 中文 L^AT_EX

第九章 中文 L^AT_EX 系统

9.1 国际化 L^AT_EX

前面第一、二两部分介绍的主要是英文 L^AT_EX。L^AT_EX 在出现之初，是完全针对于英文而设计的，有些命令中显式包含一些英文单词，例如章标题都是以 Chapter 开头，而参考文献的前面则有 Reference 或 Bibliography 等英文单词，这一切都是固化在 L^AT_EX 的核心内部。后来，在欧洲有很多人对于 L^AT_EX 做了更新，使其适合于自己的母语。例如，德国人可以在源文档中使用 `german.sty` 宏包（当时称样式文件），这样就可以生成一篇德语文章，而且章标题就自动以德语单词开头。另外，它也改变了许多其他英文中与德语规则不同的地方，例如，日期的格式。

`german.sty` 宏包的关键就是对标准 L^AT_EX 文档样式（类）进行更新，输出中的显式英文单词已被名称命令代替。为了适用于所有语言，在欧洲用户间这些名称已经被标准化了。来自 Darmstadt 的 J. Schrod 对 L^AT_EX 版本进行了修改，得到了著名的 L^AT_EX，即国际化的 L^AT_EX。

到了 1991 年 12 月 1 日，这些名称特征成为 L^AT_EX 标准的一部分。通过这种方法，即使是英语用户也很容易改变某些标题，例如，把 ‘Abstract’ 改成 ‘Summary’，或者把 ‘Contents’ 改成 ‘Table of Contents’。

9.2 常用的中文 T_EX 系统

做为世界上使用者数目最多的语言，中文也应当可以与 L^AT_EX 很好地协调工作。这也是很多以中文为母语的 L^AT_EX 高手追求的目标。现在，已有许多的中文 T_EX 或 L^AT_EX 系统。

9.2.1 ChT_EX

这是王景白开发的一个中文 T_EX 转化程序，出现在 1989–1990 年左右。

其处理方法是源文件写好后即可用 ChT_EX 作预处理。ChT_EX 将产生一个英文 T_EX

能识别的文件,并同时产生中文字体定义头文件(header file)。接着便可由 T_EX 将其变为 .dvi 文件。最后用 dvi2ps 将 .dvi 文件翻译成 PostScript 文件,这样便可直接打印。这一过程虽然复杂,但如果借助了批处理功能,是可以很快完成的。

ChT_EX 利用 T_EX 的特点,将源文件中所用的不同中文字分成几组由 218 个字母组成的字体组(font groups),并用一系列 TFM(T_EX font metric)文件来定义各字母的尺寸。对 T_EX 而言,只要 TFM 文件有定义的字,它一概放行。至于哪个是中文字哪个是英文字,那就靠 ChT_EX 和 dvi2ps 来合作了。ChT_EX 用了一个具有约 7000 字的 24×24 点阵简体字字库,以通讯用汉字字符集(基本集)国家标准(GB2312-80)为母本。ChT_EX 在 T_EX 中是用 METAFONT 排版中文字符,而且可以排版一段中文。但是中文字符的 METAFONT 字体数目是很有限的。

现在流行的 CCT 中文 L^AT_EX 与之原理非常相似。

9.2.2 Cc2tex

Cc2tex 是一个可以帮助用户以 L^AT_EX 格式输入中文字符的工具,由新加坡的 Zhi Biao Wu 开发,最近的版本也是早在 1992 年出现的 1.3 版。

这个工具根据用户命令,首先把汉字组织到一个 PostScript 文件中,然后用 psfig 命令把这个文件包含到 L^AT_EX 中。这样我们就可以成功地在—行内排出汉字。但由于实现方法的限制,无法生成长的中文文章。因此开发者干脆称之为伪中文 L^AT_EX。据开发者称,当初他并不知道在网络上可以得到 ChT_EX,而且他也希望能早些开发出真正的中文 L^AT_EX,从而使自己的系统失去作用。

9.2.3 ChiT_EX 和 ChiL^AT_EX

ChiT_EX 和 ChiL^AT_EX 是由台湾的陈弘毅开发的繁体字中文 Plain T_EX 和 L^AT_EX。可以应用于 Unix 上的 T_EX 和 Windows 平台上的 Scientific WorkPlace。

编写 ChiT_EX 始于为 Unix 系统开发的一个中文 T_EX 处理程序。第一个版本出现在 1994 年 2 月,现在版本为 6.0(1997 年 10 月)。在 1995 年 8 月,出现了 ChiL^AT_EX,即建立在 ChiT_EX 上的中文 L^AT_EX。

ChiT_EX 和 ChiL^AT_EX 可以利用 Adobe Type 和 True Type 字体。在处理时,把源文件保存为 .tex 文件,用 chilatex 编译处理源文件。

9.2.4 CJK 宏包

CJK 宏包是专门为 L^AT_EX 2_ε 设计的一个样式宏包,可以处理中文、日文和韩文。该宏包由 Werner Lemberg 设计,版本为 4.1.3(20-Jun-1997)。

要使用该宏包,需要用 1996 月 12 月 1 日发行的 L^AT_EX 2_ε 或者更新的版本。在源文件中,使用格式为

```
\documentclass{article}
\usepackage{CJK}
```

这样我们就拥有了两个新环境,一个是

```
\begin{CJK}[ 字体编码 ]{ 编码 }{ 族 }
...
\end{CJK}
```

另一个是星号形式的环境

```
\begin{CJK*}[ 字体编码 ]{ 编码 }{ 族 }
...
\end{CJK*}
```

其中各参数的意义为:

编码 指的是目前在 CJK.enc 文件中实现了的编码, 它们有:

- Bg5(Big5),
- GB(简体国际, GB 2312-80),
- GBt(繁体国标, GB 12345-90)
- JIS(Japanese Industry Standard, JIS X 0208-1990)
- JIS2(JIS supplementary character set, G1 = JIS X 0212-1990)
- SJIS(Shift-JIS aka MS-kanji)
- KS(Korean Standard hangul and hanja, KSC 5601-1987)
- UTF8(UTF 8 (Unicode Transformation format 8), 也称为 UTF 2 或 FSS-UTF)
- CNS1(Chinese National Standard Plane 1, CNS 11643-1992 plane 1)
- CNS2...CNS7(plane 2 - 7)
- CEFX(专为 IRIZ 的 CEF 编码保留)
- CEFY(CEF 编码专用)

字体编码 现在已定义的字体编码为: “(空, 默认值)、‘pmC’ (适用于 Big5, GB, GBt, JIS 和 KS), ‘dnp’ 和 ‘wn’(适用于 JIS), ‘HL’(适用于 KS)。

族 如果这个参数为空, 就选择在 CJK.enc 给出的默认值: 所有的编码都是 ‘song’, 而 KS 的族为 ‘mj’

CJK* 环境会去掉在 CJK 字符后面的没有保护的空格和新行, 这也是中文和日文的习惯。而 CJK 就不会做到这点, 这适用于欧洲语言和韩文。

下面就是一个典型的示例:

```
\begin{CJK*}{GB}{kai}
  要以国标编码显示的文本
\end{CJK*}
```

9.2.5 综述

从上面几种系统的介绍可见, 由于标准 TeX 软件不能直接处理含有汉字的 TeX 源文件, 因此中文 TeX 系统通常用一个预处理软件把中文源文件转化为 TeX 文件或者图形文件, 然后再用 TeX 软件进行编译, 生成排版结果 DVI 文件。由于这种 DVI 文件中包含了汉字信息, 因此中文 TeX 系统一般还提供了调用汉字字库显示以及打印 DVI 文件的程序; 或对 DVI 文件进行转化之后再用标准 DVI 驱动程序 (如 dvi2ps.exe 等) 来进

行输出。

目前国内常见的两种中文 T_EX 的系统是：由天元基金支持开发的天元系统，以及中国科学院计算数学与科学工程计算研究所张林波开发的 CCT 中文 L^AT_EX 系统。由于笔者起初接触中文 L^AT_EX 时，使用的就是 CCT 系统，因此我们在下一节把注意力放在这个系统上。因为我们认为，这是一个非常成功的中文 L^AT_EX 系统。如果读者只是想知道如何使用 CCT 中文 L^AT_EX，就不必阅读下面的内容，直接学习第十章的内容。

9.3 CCT 系统简介

emT_EX 是现在 MS-DOS 上最流行的免费 T_EX 实现版本，张林波所提供的 CCT 中文 L^AT_EX 就是以 emT_EX 为基础的。

现在 emT_EX 的版本为 3.14159[4a]，它既支持 L^AT_EX 2_ε，也支持在不久的将来会被淘汰的 L^AT_EX 2.09。这个系统的功能完整实现了所有 T_EX 和 L^AT_EX 以及 A_MS-L^AT_EX 的功能。CCT 中文 L^AT_EX 系统是在 emT_EX 的基础上进行了适当的配置，增加了与处理中文有关的命令和环境，提供了多种中文字体，另外还设计了许多辅助程序。

CCT MSDOS 版运行所需要的环境为：MSDOS 3.0 以上版本，任何基于 GB2312 编码的中文编辑系统（用于汉字的输入和编辑），以及 2.0 以上版本的 T_EX 系统。这种系统要求是非常低的，计算机软硬件发展到今天，满足这种条件是轻而易举的。

CCT 系统的最新 DOS 版本可从网址 <ftp://ftp.cc.ac.cn/pub/cct/msdos/> 获得。

9.3.1 CCT 系统的构成

CCT 系统由下列文件组成：

初始化程序：	CCTINIT.EXE
预处理程序：	CCT.EXE
屏幕显示驱动程序：	DVISCR.EXE
24 针打印机驱动程序：	DVI24P.EXE (也用于驱动 Canon 喷墨打印机)
HP 激光打印机驱动程序：	DVILJP.EXE (适用于 HP LaserJet+ 激光打印机)
北佳激光打印机驱动程序：	DVIPECAN.EXE (适用于北佳视频卡)
喷墨打印机驱动程序：	DVIDJ500.EXE (适用于 HP DeskJet 系列喷墨打印机)
字号定义文件：	CCT.DAT
字库定义文件：	CCFONTS.DEF
绘图程序：	PICTEX.EXE, PDFTOBFM.EXE
图象图形接口程序：	HPGL2CCT.EXE, IMG2CCT.EXE, BMF2TIF.EXE, BMF2PCX.EXE
用户拼字程序：	PZ.EXE 及有关文件

此外，CCT 系统还提供了一些非常有用的辅助程序，包括：MAKEPK.EXE, PATCHDVI.EXE, CDVIA.EXE, TEXT2DVI.EXE, UNCCT.EXE, BMFTEXT.EXE, FORALL.EXE。这些程序均带有简单的使用说明，用户从文件名及程序输出的信息中很容易知道它们的功能与用法。我们在后

面也会介绍其中几个的功能。另外, `cctwin16.exe` 或 `cctwin32.exe` 可以在 Windows 下面查看中文 .dvi 文件。

9.3.2 正确配置 CCT

在使用 CCT 系统之前, 我们首先需要安装这个程序, 然后进行必要的配置。注意, 如果不进行任何配置, 处理一般的源文件不会有什么问题, 但是谁也无法保证缺省配置总是满足你的需要。

`CCTINIT.EXE` 程序用来初始化 CCT 系统。它的作用是根据字号定义文件 `CCT.DAT` 中所定义的各个字号的大小, 生成 \TeX 排版时需要用到的 TFM 文件 (\TeX Font Metric file) 以及宏包文件 `CCHEAD.STY`。在每次修改了 `CCT.DAT` 中的参数后必须运行一次此程序以保证整个系统的正常运行, 另外首次使用 CCT 系统前最好也运行一下这个程序。

`CCTINIT.EXE` 的用法如下:

`cctinit` [可选项]

其中的方括号代表可以省略的部分 (方括号本身不用给出), 称为命令行可选项, 用来改变或设置程序运行时的一些参数。每个可选项的第一个字符必须是 “-” (减号), 后面紧跟一个字符 (字母不区分大小写) 给出可选项的名称。有些可选项的后面还带有参数。“-”与可选项名及参数之间不能有空格。不同的可选项之间必须用空格隔开。对于没用可选项给出的参数, 程序通常自动选取一个缺省值。

例如命令:

```
cctinit -t\text\texinput -x
```

中包含两个可选项 “-t” 和 “-x”, 可选项 “-t” 带有参数 “\text\texinput”。

`CCTINIT.EXE` 有下面一些可选项:

- T 后面必须跟随一个字符串参数给出放置程序产生的 TFM 文件的路径名。缺省值为 CCT 系统所在子目录 “\FONTTFMS”
- H 后面必须跟随一个字符串参数给出放置程序产生的 `CCHEAD.STY` 文件的路径名。缺省值为 CCT 系统所在子目录 “\INPUTS”
- X 没有参数。给出此可选项时, CCT 利用 \TeX 的 `\special` 指令来处理汉字的字体, 而无此可选项时, CCT 就利用一个虚拟字库 “`CCDUMMY.TFM`” 来处理汉字的字体。两种形式对用户是等效的。

例如指令:

```
cctinit -hc:\tex\macros -tc:\tex\fonts
```

指示 `CCTINIT.EXE` 将生成的 `CCHEAD.STY` 文件放在 C: 盘的子目录 “\TEX\MACROS” 中, 而将生成的 TFM 文件放在 C: 盘的子目录 “\TEX\FONTS” 中 (这些子目录最好事先建好)。

正常情况下用户不必给出可选项, 直接使用 `CCTINIT.EXE` 设定的缺省值即可。

“-H” 和 “-T” 可选项中的参数还可分别通过环境变量 `TEXINPUTS` 和 `FONTTFMS` 来进行设置 (见下面的解释)。

用户也可以通过初始化文件 `CCTINIT.INI` 来设置参数值, 例如缺省情形中在 `em \TeX` 目录中 `CCTINIT.INI` 的内容为

```
-TC:\emtex\tfm
-HC:\emtex\texinput
```

我们可以根据自己的需要编辑这个文本文件。这样做的好处就是把配置记在文件中，没有必要记住自己前一次到底用的是哪些参数。

9.3.3 环境变量的设置

在运行 emT_EX 时，需要一些环境变量，它们是：

```
EMTEXDIR  emTEX 的根目录
EMTEXOPT  emTEX 选项
TEXTFMT   标准 TEX 格式文件所在的目录
TEXTFM    TFM 文件的目录
TEXINPUT   输入文件 (如类和宏包) 的目录
TMP        临时性文件的目录
```

由上可见，变量主要包括两类，一类是目录变量，另一类是选项变量。要设置上述变量，可以采用如下形式。这里以 EMTEXDIR 为例。

```
set EMTEXDIR=d:\local\tex
```

我们在下面两节将详细讲解如何设置目录变量以及选项变量。这个设置应当放在一个批处理文件中，每次启动系统时自动调用它。例如，我们可以把它放在 emT_EX 目录中的 setenv.bat 文件中，然后在 autoexec.bat 文件中调用该批处理文件。

9.3.4 emT_EX 中的目录结构处理

在 emT_EX 编译处理源文件时，如果要查找一个文件，它就按如下步骤操作：

- 1a 如果文件中没有目录，即名称中没有：/ \ 等字符，那就在当前工作目录中寻找 (对于 TFM 文件，不执行这一步。)
- 1b 如果文件名中有目录，即包含：/ \ 等字符，那就只在指定目录中寻找，不再进行第 2 步的操作。

在执行完第 1a 步后，依据各种环境变量，执行而且只执行下述步骤中的一步：

- 2a 如果设置了相应环境变量 (例如 TEXINPUT)，那么就按指定的顺序，搜索变量中所有的目录。对每个目录，可以按后面讲的方法，进行子目录搜索。目录间用 ‘;’ 分开。
- 2b 如果没有设置相应环境变量，但设置了 EMTEXDIR 环境变量，那么就搜索由 EMTEXDIR 指定的目录中的一个默认目录以及该默认目录的一级子目录。系统使用下述默认目录：

```
data      相应于 TCP 文件
texinput   相应于输入文件
texfmts    相应于标准 TEX 的格式文件
tfm        相应于 TFM 文件
```

例如，如果没有设置 TEXINPUT，而 EMTEXDIR 设为 d:\tex，那么就在 d:\tex\texinput 及其一级子目录中搜索输入文件。

2c 如果相应的环境变量以及 EMTEXDIR 都没有设置, 那么就搜索 `\emtex` 下的默认目录及其一级子目录。例如, 就在 `\emtex\texinput` 及其一级子目录中搜索输入文件。

如果所用的计算机上只有一个驱动器, 那么就不一定需要设置环境变量。

但是如果把 `emTeX` 安装到其他名称的主目录下面, 而没有改变子目录的名称, 或者计算机中有多个驱动器, 则只要设置 EMTEXDIR 也就足够了。例如, 若 `emTeX` 安装在 `d:\local\tex`, 那么就需要用

```
set emtexdir=d:\local\tex
```

设置 EMTEXDIR, 这样就会在 `d:\local\tex\texinput` 中搜索输入文件, 在 `d:\local\tex\tfm` 中搜索 TFM 文件, 其他类似。

如果对目录结构进行了修改, 那就必须设置类似于 TEXINPUT 等各个环境变量。这时设置了 EMTEXDIR, 再用其他环境变量覆盖默认目录也就可以了。假设把 `emTeX` 安装在 `d:\emtex` 目录中, 而在 `d:\tex\styles` 还有一些输入文件, 那么就可以用下述命令:

```
set emtexdir=d:\emtex
set texinput=d:\emtex\texinput!;d:\tex\styles
```

子目录的递归搜索

`emTeX` 支持子目录搜索。也就是说可以自动对指定目录的一级子目录或者所有的子目录进行搜索。在目录名称后面加上 `!` 就可以进行一级子目录搜索, 加上 `!!` 就会使 `emTeX` 对指定目录的所有子目录进行搜索。对默认目录总是会自动进行一级子目录搜索, 即

```
set texinput=\emtex\texinput!
```

等价于默认行为。

假设我们有如下的目录结构:

```
\emtex\texinput
\emtex\texinput\local
\emtex\texinput\local\joe
\emtex\texinput\local\zoe
\emtex\texinput\latex
\emtex\texinput\amstex
```

那么设置为

```
set texinput=\emtex\texinput
```

时, 只搜索 `\emtex\texinput`, 但若设置为

```
set texinput=\emtex\texinput!
```

那么就会在如下目录中搜索输入文件:

```
\emtex\texinput
\emtex\texinput\local
\emtex\texinput\latex
\emtex\texinput\amstex
```

而

```
set texinput=\emtex\texinput!!
```

会在上面列出的所有目录中进行搜索。

9.3.5 emT_EX 的内存布局

在绝大多数情形中, 由 emT_EX 所提供的缺省内存布局是非常恰当的。但是如果要把 emT_EX 应用于中文源文件的处理, 而且文件很大时, 经常会出现 TeX capacity exceeded, sorry [...=###] 这样的消息。这里中括号内的等式一方面告诉你系统当前该类内存的值为多少, 另一方面也告诉你这一内存被用完了。笔者在编写本书的过程中, 全书编译到 130 多页的时候, 就会有这样一条消息:

```
\texttt{TeX capacity exceeded, sorry [save size=1000]}
```

从而无法一次编译完全书的内容。还好, 有 `\include` 和 `\includeonly` 命令, 我们可以把每章的内容放到一个 .dvi 文件中, 最后得到的目录表的信息也是正确的。但这总不是一个令人满意的解决方法。

实际上, 我们可以利用 EMTEXOPT 变量来改变 emT_EX 的缺省内存分配方案。表 9.1 给出了各种内存的分配情况。

表 9.1 emT_EX 内存分配

选项	描 述	取值范围	缺省值
/ma	parameter stack (macro parameters)	60-1000	60
/mf	font memory (font metric data)	5000-65500	32766
/mm	main memory (main memory)	262142-1048576	262142
/mn	semantic nest size (mode nesting)	20-3000	40 (2)
/mp	pool size (strings)	20000-65500	50000
/ms	save size (values saved by grouping)	100-16000	600
/mt	pattern memory (hyphenation)	5000-65500	10000

这样我们就可以利用

```
set emtexopt=/ms:16000
```

来增加存储空间, 以解决我们前面提到的问题。一行中可以进行多个设置, 如

```
set emtexopt=/ms:2000 /mp:45000
```

9.4 CCT 处理流程

一般来说, 用 CCT 系统排版 L^AT_EX 的过程是:

1. 准备 CCT 源文件: 中文 L^AT_EX 文件本质上与英文 L^AT_EX 文件是一样的, 在 CCT 系统中, 只需要将 `\documentclass` 中的 `book` 或 `article` 类换成 `cctbook` 或 `cctart` 类 (如果使用的是 L^AT_EX 2.09 兼容模式中的 `\documentstyle`, 可以用 `cbook` 或 `carticle` 样式), 将源文件名后缀取为 `.ctx`, 即可在正文中使用汉字作为 CCT 源文件。

假设我们输入了如下一篇文档，保存为 `sample.ctx`：

```
\documentclass[11pt]{cctart}
\title{ 学术论文 }
\author{ 邓建松 \texguru@263.net }
\begin{document}
\maketitle
这是我的第一篇 CCT
中文 \LaTeXe 文件。
\end{document}
```

2. 利用 CCT 系统的 `cct.exe` 预处理程序将 CCT 源文件转化为 \TeX 源文件。对上面的示例源文件，命令格式为

```
cct sample
```

不必加后缀 `.ctx`。结果生成 `sample.tex` 文件，用文本编辑器打开这个文件，内容为

```
\documentclass[11pt]{cctart}
\title{{\CC Q' Ju B[ ND]}}
\author{{\CC 5K =( KI} \texguru@263.net}

\begin{document}
\maketitle
{\CC Ub JG NR 5D 5Z R; F*} CCT
{\CC VP ND} \LaTeXe {\CC ND <{\char126}~{\char33}{\char35}\CCA
} \end{document}
```

其含义是难以理解的，但我们在后面就要用 \LaTeX 2_ϵ 处理这个文件。

3. 用 \LaTeX 2_ϵ 对所得的 \TeX 源文件 (`sample.tex`) 进行编译处理，生成排版结果文件：

```
latex2e sample
```

就会得到 `sample.dvi` 文件。同样这里也没有必要加上后缀 `.tex`。

4. 由于我们这个示例文件非常简单，我们可以用 CCT 的驱动程序显示或打印排版结果文件，例如在 DOS 提示符后面输入

```
dvisrc sample
```

(后缀 `.dvi` 可以省略)，就可以在屏幕上显示结果。在现在的 CCT 版本中，`emTeX` 根目录中有一个 `view.bat` 批处理文件，其内容为

```
@echo off
dvisrc -gvesa:1024:768 -r300 -z2 %1 %2 %3 %4 %5
```

因此可以直接用 `view sample` 来查看 `sample.dvi` 文件。另外，我们也可以用 Windows 平台程序 `cctwin16.exe` 或 `cctwin32.exe` 来查看结果。

如果对结果非常满意，那么可以用

```
dvi24p sample    或    dvi1jp sample
```

等命令在针式打印机或激光打印机上打印出来。

5. 由于 .dvi 文件的浏览, 需要 T_EX 字体。如果想把结果送给其他人阅读, 这时我们就需要把它转化成 PostScript 格式, 从而使收到文件的人, 不需要安装 CCT 中文 L^AT_EX 也能阅读文档。另一方面, 如果在 T_EX 文件中利用 8.11 节中的方法插入了外部图形, 这时利用 dvisrc 和 cctwin 是无法看到图形的, 在 .dvi 文件中相应于图形的地方是一片空白。为此我们也需要把 .dvi 文件转化为 PS 文件。

如果我们的源文件中没有任何中文字符, 那么直接调用 dvips32.exe 程序就可以生成相应的 PostScript 文件。假设 .dvi 文件为 exam.dvi, 那么

```
dvidrv dvips32 exam.dvi
```

就会生成 PS 文件 exam.ps, 这样就可以用任何 PostScript 浏览器 (如 GhostView) 来查看。

CCT 系统中没有提供 PostScript 驱动程序。为使用户能够产生 PostScript 输出, 从 5.1 版开始新增了一个程序 PATCHDVI.EXE, 它能将包含汉字的 .dvi 文件转换成普通 .dvi 文件, 并把其中的汉字组织成几个 PK 文件, 这样用户可以使用任何 DVI 驱动程序 (包括 PS 驱动程序) 进行处理。事实上, 这也是最初的 CCT 版本的工作方式。因此对我们的 sample.dvi 文件, 可以用

```
patchdvi -y sample sample1.dvi
```

来把它转化成普通 .dvi 文件。然后与前面一样, 用

```
dvidrv dvips32 sample1.dvi
```

可以得到 sample1.ps 文件。

我们建议使用 emT_EX 中包含的 DVIPS.EXE 程序来产生 PostScript 输出, 在

<http://202.38.68.78/~texguru/software/dvips.zip>

处可下载该程序。

为了简化上述操作, 在 T_EXGuru 上提供了一个非常简单的批处理文件, 内容为:

```
@echo off
if exist %1.ctx call cct %1
call latex2e %1
call patchdvi -y %1.dvi ~~tmp.dvi
rem call cdvia %1.dvi ~~tmp.dvi
dvidrv dvips32 ~~tmp.dvi
if exist %1.ps del %1.ps
ren ~~tmp.ps %1.ps
del ~~tmp.dvi
echo on
```

假设这个批处理的文件名为 tp.bat, 那么

```
tp sample
```

就会从 `sample.ctx` 生成 `sample.ps`，当然这里假定源文件中没有错误，可以顺利地编译生成 `.dvi` 文件。在调用这个批处理文件时，参数一定不能有扩展名。而且如果没有 `.ctx` 文件，只有 `.tex` 文件，用它也能生成 PS 文件。因为批处理文件在第一步中做了判断，确定是否有必要用 `cct.exe` 进行转化。

另外在 `TEXGuru` 上的 `dvips` 是假设 `emTEX` 安装在 `C:\EMTEX` 目录中，这时只要把 ZIP 文件解开，相应的文件会各就其位。如果 `emTEX` 安装在其他目录中，需要参考 `TEXGuru` 主页中的修改方法。

`PATCHDVI` 程序是一个 DOS 下的命令，其命令格式为

```
patchdvi [选项] 输入文件 [.dvi] [选项] 输出文件 [.dvi] [选项]
```

其中 选项 的值为

`-y` 所有问题都给出肯定的回答，这样程序不会中途停下来等用户做出反应。

`-r<dpi>x<dpi>` 指定分辨率（默认值：`-r300x300`）。

`-m<mag>` 改变 `.dvi` 文件的全局放缩因子。

`-f` 不为中文字符创建 PK 字体。

`-b, -b0` 利用

```
\special{em:graph file.pcx ####in ####in}
```

代替

```
\special{BMF=file.bmf}(xdvi, emTEX)。
```

`-b1` 利用

```
\special{isoscale file.pcx #####in #####in}
```

代替

```
\special{BMF=file.bmf}(windvi)。
```

`-q` 让程序“无声”的执行下去，不显示消息。

`-p<str>` 利用 `str` 做为 PK 字体名的前缀，默认值为 `tmp`。

`PATCHDVI` 程序生成的 PK 字体放在 `dpi888` 子目录中。

第十章 CCT 中文 L^AT_EX 的使用

10.1 CCT 源文档的结构

下面是 CCT 中文 L^AT_EX 源文档的典型结构, 可以把它做为自已开始使用 CCT 的模板。

```
\documentclass[11pt]{cctart}
\usepackage{latexsym}
\usepackage{amsmath}
\usepackage{amssymb}

\setlength{\parindent}{2\ccwd}
\setlength{\parskip}{3pt plus1pt minus2pt}
\setlength{\baselineskip}{20pt plus2pt minus1pt}
\setlength{\textheight}{21true cm}
\setlength{\textwidth}{14.5true cm}

\title{ 学术论文 }
\author{ 邓建松 \texguru@263.net }

\begin{document}
\maketitle

这是我的第一篇中文 \LaTeX{} 文档。

\end{document}
```

如果想写作中文稿件, 要用 `cctart` 或 `cctbook` 替换 `\documentclass` 命令中的 `article` 或 `book`, 这两个类分别相应于中文的短文和书籍。

`cctart` 类与 `article` 类基本一样, 但对目录、插图目录、表格目录、参考文献、索引、图表、摘要、部分以及附录的名称进行了汉化, 例如 `thebibliography` 环境前面的名称从 `Reference` 变为中文的“参考文献”

同样, 等价于英文 `book` 类的 `cctbook` 也对上述名称进行了汉化, 另外每章的标题自动显示为“第二章”等形式。

所有普通英文 L^AT_EX 2_ε 中可以使用的命令, 在 CCT 中文 L^AT_EX 2_ε 中也可以使用, 而且功能基本一致, 如果有差别的话, 那也是针对中文习惯进行了很细微的修改。例如上面的名称汉化处理。

另外, 在 CCT 系统中还增加了许多新的命令和参数。

10.2 CCT 系统增加的命令

如果用户在 CCT 源文件开头加入了 `\usepackage{cchead}` 或者 `\input cchead.sty` (`cctart` 或 `cctbook` 类内部调用了 `cchead.sty`, 因此上面的模板没有再显式调用它), 那么除了可以使用标准 \TeX/L\TeX 命令外, 还可以使用 CCT 增加的与汉字有关的一些命令:

10.2.1 `\zihao`

`\zihao{}` 用来选择汉字大小, 但它不改变西文字号的大小, 而西文字号的改变可以改变汉字字号的大小, 花括号的参数为选用的字号。各字号大小由 CCT 系统的 `cct.dat` 文件定义, 缺省情形中用户可以使用以下十种字号:

CCT 指令	对应的字号	CCT 指令	对应的字号
<code>\zihao{0}</code>	初号	<code>\zihao{-4}</code>	小四号
<code>\zihao{1}</code>	一号	<code>\zihao{5}</code>	五号
<code>\zihao{2}</code>	二号	<code>\zihao{-5}</code>	小五号
<code>\zihao{3}</code>	三号	<code>\zihao{6}</code>	六号
<code>\zihao{4}</code>	四号	<code>\zihao{7}</code>	七号

CCT 的程序运行时从 `cct.dat` 文件中得到有关汉字大小、间距等信息。用户可以通过修改此文件来设置每种字号的尺寸。如果修改了这个文件, 则需重新运行一次 `CCTINIT.EXE` 程序来生成相应的 `.TFM` 文件及 `cchead.sty` 文件。 `cct.dat` 必须与 `CCTINIT.EXE` 位于同一子目录中。

`cct.dat` 是一个普通文本文件。它的开头是一个正整数, 给出可以同时使用的不同字号的个数, 最大不能超过 26 (因此用户最多只能同时使用 26 种不同大小的字号)。接着顺序给出定义每个字号的数据。每个字号由 5 个数定义, 其中前 4 个数是浮点数, 而第 5 个数为整数。这些数的含义如下:

- 第一个数: 字宽 (以 pt 为单位)
- 第二个数: 字高 (以 pt 为单位)
- 第三个数: 字距与字宽之比
- 第四个数: 行距与字高之比
- 第五个数: 给出 `\zihao` 指令中应该使用的参数 (即字号)

不同数据之间需用空格或换行隔开。

下面给出的是 CCT 系统配置的原始 `cct.dat` 文件中的数据, 它定义了十种不同大小的字号:

```

10
34.0    34.0    0.06    0.2    0
26.0    26.0    0.06    0.2    1
20.0    20.0    0.06    0.2    2

```

15.0	15.0	0.06	0.2	3
13.0	13.0	0.06	0.2	4
11.32	11.32	0.06	0.2	-4
9.9	9.9	0.06	0.2	5
8.5	8.5	0.06	0.2	-5
7.5	7.5	0.06	0.2	6
5.0	5.0	0.06	0.2	7

例如第二行数据说明 `\zihao{0}` (初号字) 的大小为: 字宽 = 34pt, 字距 = $34 \times 0.06 = 2.04\text{pt}$, 字高 = 34pt, 行距 = $34 \times 0.2 = 6.8\text{pt}$. 由于其中定义的字号属于标准字号, 因此用户最好不要修改它们的大小, 若想使用其他大小的字号可以加在它们后面.

值得注意的是, `cct.dat` 文件中所定义的大小不是绝对的. 如果用户在 CCT 源文件中或输出结果时使用了 “`\magnification`” 参数, 则所有的汉字也会相应地被放大或缩小.

10.2.2 `\ziti`

`\ziti{}` 用来设置汉字的字体, 花括号中的参数为字母. CCT 系统在缺省情况下将 `\ziti{A}`, `\ziti{B}`, `\ziti{C}`, `\ziti{D}`, `\ziti{E}` 分别设为宋体、黑体、楷书体、仿宋体和标宋体, 他们分别可以简记为 `\songti`, `\heiti`, `\kaishu`, `\fangsong`, `\biaosong`. 用户也可以增加 `ccfonts.def` 中定义的字库文件名, 并使用 `\ziti{F}`, `\ziti{G}` 等来选择其他字体. 例如: 下述指令

```
\zihao{5}\ziti{A} 这是宋体五号, \kaishu\zihao{-5} 这是楷书体小五号,
\songti 这是宋体小五号
```

的输出为:

```
这是宋体五号, 这是楷书体小五号, 这是宋体小五号
```

字库定义文件 `ccfonts.def` 是一个只被 CCT 的设备驱动程序用到的正文文件. 它定义汉字字库的文件名.

CCT 允许用户最多同时使用 26 种字体. 每种字体被分为两级, 分别存贮在两个字库文件中. 第一级字库包括区位表中第 16 区到第 55 区中的汉字, 第二级字库包括第 56 区到第 87 区中的汉字. 另外还有一个文件包含区位表中第 1 区到第 15 区中的符号和一个用户字库文件. `ccfonts.def` 文件中按下面的顺序给出这些字库文件的文件名:

```
用户字库
标点符号库 (1-15 区)
第一种字体 (\ziti{A}) 的一级字库名
第一种字体 (\ziti{A}) 的二级字库名
第二种字体 (\ziti{B}) 的一级字库名
第二种字体 (\ziti{B}) 的二级字库名
....
```

每个字库文件名后面还必须跟随一个比例因子用来调整该字库的相对大小. 字库文

件名与比例因子之间必须用空格或换行隔开。由于驱动程序忽略这些数据之后的内容，用户可在最后一个字库文件名和比例因子之后加入自己的注解或说明。

下面是一个字库定义文件的例子，其中定义了宋体、黑体、楷书、仿宋和标宋五种标准字体的字库名。

```
USERFONT.PZ    1.0
CLIB256E.PPS   1.0
CL256S0.PPS    1.0
CL256S1.PPS    1.0
CL256H0.PPS    1.0
CL256H1.PPS    1.0
CL256K0.PPS    1.1
CL256K1.PPS    1.1
CL256F0.PPS    1.1
CL256F1.PPS    1.1
CL96BS0.PPS    1.0
CL96BS1.PPS    1.0
```

它说明 1-15 区的符号在文件 CLIB256E.PPS 中，一级宋体字库在文件 CL256S0.PPS 中，二级宋体字库在文件 CL256S1.PPS 中，用户字库文件为 USERFONT.PZ，等等。其中仿宋体和楷体输出时将被放大 1.1 倍。

如果需要使用五种基本字体之外的字体，可将相应的字库文件名加在上述五种基本字库文件之后，并在排版源文件中利用 \ziti 命令来选用相应的字体。

比例因子限制在 0.5 到 2 之间。如果所给出的比例因子小于等于 0.5 或大于等于 2，则它将被忽略。

CCT 5.13 版的驱动程序在调入用户字库时，首先在 DVI 文件所在的目录中寻找与 DVI 文件同名，扩展名为 .PZ 的文件，如果该文件存在，则将该文件做为用户字库调入，当该文件不存在时才使用 ccfonts.def 中定义的用户字库。这样做可以方便用户在不同的 DVI 文件中使用不同的用户字库。

我们在 10.5 节讨论了如何在 CCT 系统中使用 True Type 字体的方法。

10.2.3 \ziju

有时用户希望在同一篇文章中使用不同的字距和行距。通过在 CCT 源文件中适当设置 TeX 的 \baselineskip, \lineskip 和 \lineskiplimit 等变量，用户可以得到任意的行距。

\ziju{} 用来改变缺省的汉字字距，花括号中参数为字距和字宽的比，0.06 是缺省值。

例如：下述指令

缺省字距为 0.06，\ziju{0.6} 调整字距为缺省值的 10 倍的输出为：

缺省字距为 0.06， 调整字距为缺省值的 10 倍

但这条指令对西文字母和单词的间距没有作用。

10.2.4 `\ccwd`, `\ccht`, `\ccdp`

`\ccwd`, `\ccht`, `\ccdp` 这三个长度单位分别表示当前字号的字宽 + 字距、字高和字深。其中字高是指一个字符在其所在行的基线以上部分的高度, 而字深是指基线以下的部分。通常可以使用 `\setlength{\parindent}{2\ccwd}` 将段头缩进设为两个汉字的宽度 (字宽 + 字距), 这样就完全符合中文的习惯。

10.2.5 `\pushziti`, `\popziti`

`\pushziti` 将当前字体压入栈中 (栈的最大深度为 16), 而 `\popziti` 则将当前字体恢复为上次 `\pushziti` 保存的字体。这两条指令是用于在 T_EX 的浮动对象 (如页眉、页脚、脚注以及 L^AT_EX 的 `table` 和 `figure` 环境) 中, 用来防止改变其中的字体而影响紧跟浮动对象之后的汉字字体的。

例如设置页眉时如果需要改变页眉字体, 而要避免正文使用的不同字体受页眉的影响, 我们可以如下定义

```
\markboth{\pushziti 奇数页眉 \popziti}{\pushziti{ 偶数页眉 }\popziti}
```

10.2.6 `\ccnospace`, `\CS`

`\ccnospace{}` 用来取消 CCT 排版中加在中文和西文字符之间的空档, `\CS` 是用来留出相当于两个汉字之间字距的空档。

这两条命令的定义为:

```
\def\CS{{\CC\ }}
\def\ccnospace#1{\leavevmode\unskip #1\ignorespaces }
```

具体使用见 10.3.2 节的介绍。

10.2.7 `\chntoday`

`\chntoday` 定义了当前日期的中文表示。例如 `\chntoday` 的输出就有可能为 “2001 年 8 月 12 日”。这条命令的定义是保存在 `ccbase.sty` 中, 其具体定义为

```
\newcommand\chntoday{\the\year\ {\CC Dj} \the\month\ {\CC TB} \%
\the\day\ {\CC HU}}
```

这实际上是

```
\newcommand\chntoday{\the\year\ 年 \the\month\ 月 \the\day\ 日 }
```

用 `cct.exe` 翻译后的结果。缺省方式下, 文章的 `\date` 使用的就是 `\date{\chntoday}` 格式。

10.2.8 `\chnno`

`\chnno` 定义了数字的汉字表示。它的使用与 `\arabic` 或 `\roman` 等命令类似。

例如: 如果使用下述输入,

```
第 \ccnospace{\CS\chnno{chapter}\CS} 章第 \arabic{section} 节
```

就得到“第十章第2节”。注意,这条命令只接受记数器做为参数。如果输入了`\chnno{0}`,就会给出错误信息。

为了直接把数字翻译成中文,可以利用`\chnno@one`和`\chnno@two`命令来翻译一位数或两位数。显然,要把命令放在`\makeatletter`和`\makeatother`之间。例如:

```
\makeatletter
\chnno@one{3} 或 \chnno@one4 及 \chnno@two21
\makeatother
```

来表示中文数字“三”或“四”及“二十一”。

对于后两条命令,由于其只是内部命令,因此尽量少使用。必要时,可以临时创建一个记数器,使用`\chnno`命令。这里的两条内部命令也没有考虑很多的特殊情形。例如,`\chnno@two01`的结果为 零十一。

10.3 精调中文排版

10.3.1 章节命令的使用

在排版中文时,我们仍然要像在通常的西文 \LaTeX 2_ϵ 中那样,使用章节命令,因为在`cctbook`和`cctart`中已对这些命令的名称参数进行了汉化。例如,在`cctbook`中包含如下定义:

```
\renewcommand\contentsname{目 录}
\renewcommand\listfigurename{插图目录}
\renewcommand\listtablename{表格目录}
\renewcommand\bibname{参考文献}
\renewcommand\indexname{索 引}
\renewcommand\figurename{\rm 图}
\renewcommand\tablename{\bf 表}
\renewcommand\partname{第 \ccnospace{\CS\chnno{part}}\CS} 部分}
\renewcommand\chaptername{第 \ccnospace{\CS\chnno{chapter}}\CS} 章}
\renewcommand\appendixname{附录 \ccnospace{\CS\chnno{chapter}}}
```

这样,我们就不必由于担心出现了英文`Chapter`,而放弃对`\chapter`命令的使用。因为充分利用这些章节等命令,可以在生成目录表等方面节省很多时间。

10.3.2 汉字与西文之间的空档

假设CCT的源文件中含有下面的内容:

```
CCCCCGeEEEECCCCCCCC
```

其中`C`代表汉字, `e`代表其他(非汉字)字符,则CCT的预处理程序将其转换成:

```
{\CC .....} eeee {\CC .....}
```

其中“...”代表汉字转换后产生的字符。这样的处理使得在汉字与西文字符间有相当于一个西文空格的间距。在用户排版源文件时，中西文之间留不留空格都不影响排版输出的结果。但有时用户不希望有多余的空格。例如下面的一段排版命令：

```
啊啊啊 \hspace{0pt} 啊啊
```

它的排版输出为：

啊啊啊 啊啊

其中出现了一个多余的空格。目前 CCT 的处理方式尚无法自动避免这个问题。做为一个解决方案，我们建议使用 `\CS` 和 `\ccnospace` 命令。其中前者留出相当于两个汉字之间字间距的空档，而 `\ccnospace` 则用来取消多余的空档。将上例中的排版命令改成：

```
啊啊啊 \ccnospace{\CS\index{abc}} 啊啊
```

则输出变为：

啊啊啊啊啊

即保证了汉字字间距的均匀性。注意上例中使用了“`\CS`”命令以留出汉字间的正常字间距。

对于一些需要大量在汉字中间使用面又不希望引入多余的空格的命令，可引进适当的宏定义以简化排版。例如，如果需要大量使用 `\index` 命令，则可引进定义：

```
\def\cindex#1{\ccnospace{\CS\index{#1}}}
```

在两个汉字之间使用 `\cindex` 命令，而在其他情况（前或后为西文时）仍使用 `\index` 命令，便可避免在插入 `\index` 命令的地方出现多余的空格。

如果插入的内容由汉字构成，当用 `\ccnospace` 命令去除多余空格时，还需注意在前后留出汉字的正常字间距。请看下例：

```
\ziju{0.5} 宋体宋体 \ccnospace{{\heiti 黑体}} 宋体宋体
```

这里为了看得更清楚，我们有意使用了大的字间距。上述排版命令的输出如下：

宋 体 宋 体 黑 体 宋 体 宋 体

注意“黑体”前后的空档被 `\ccnospace` 命令吃掉了。上述排版命令应该写成如下形式：

```
\ziju{0.5} 宋体宋体 \ccnospace{\CS{\heiti 黑体}\CS} 宋体宋体
```

才能得到正确的输出结果：

宋 体 宋 体 黑 体 宋 体 宋 体

还有一点要注意的是与西文排版一样，`TEX` 命令后面的空格不起作用，因此必要时需要注意命令的写法。如应该将

中文 `\TeX` 排版系统

写成

中文 `\TeX\` 排版系统

具体排版过程中，用户可根据情况灵活使用上述技巧来保证排版质量。最好结合宏定义来使用上面介绍的命令，使得排版源文件更加简洁。

10.3.3 中文排版的断行处理

中文排版与西文排版一样，对于在何处断行有所限制，排版术语中称为“行禁则”。中

文排版的行禁则主要与标点符号有关,如“,”、“;”等之前不应断行,“(”之后不应断行,等等。无论是中文还是西文,用户排版时都应该注意不要在不应该断行的地方输入多余的空格。只要做到这一点,CCT基本上可以保证排版输出符合行禁则。必要时也可使用 \LaTeX 的“\nolinebreak”命令(当在汉字中间时应该写成“\ccnospace{\CS\nolinebreak}”的形式来避免引进多余的空格)或“\mbox”命令来禁止在某处断行。

10.3.4 西文标点符号的转换

一些用户在输入标点符号时习惯使用西文标点符号(半角符号),而有的则喜欢使用中文标点符号(全角符号)。为了保持排版输出的一致性,CCT的预处理程序自动将一些紧跟在汉字后面的半角符号转换成全角符号。这样的符号有:“!”,“,”,“.”,“:”,“;”,“?”。通常这样处理不会产生问题,事实上许多用户从未注意到这种转换。但有一些特殊情况也须加以注意。例如,\index命令中用字符“!”来表示次级索引,因此使用\index命令生成索引时可能会有下面形式的排版命令:

```
\index{\LaTeX 程序设计 !\LaTeX 程序设计命令}
```

其中的字符“!”用来指明“ \LaTeX 程序设计命令”是一个二级索引。但由于“!”跟在汉字的后面,因此CCT的预处理程序会将它转换成全角字符。此时用户需在“!”的前面插入一些不影响排版的字符将它与汉字隔开来避免发生问题。例如,可将上面的排版命令写成:

```
\index{\LaTeX 程序设计 {}!\LaTeX 程序设计命令}
```

或

```
\index{\LaTeX 程序设计 \relax!\LaTeX 程序设计命令}
```

此外在用“CCT.EXE”处理时还应该使用“-s33 -s34 -s124”选项,避免索引中的汉字码转换成字符“!”,“”和“|”(参看10.4节)。

10.3.5 标点符号与汉字的对齐

CCT排版时会将一些标点符号前面或后面的空减少一些,使它们的宽度比一个汉字小。但有些时候用户希望标点符号与汉字的宽度一样以便对齐,为此只需使用下面形式的排版命令:

```
{
  \def\CCA{\relax} \def\CCAS{\relax}
  \def\CCB{\relax} \def\CCBS{\relax}

  排版内容 ... ..
}
```

10.3.6 用字数控制距离

有时用户希望用汉字的字数为单位来进行度量。例如为了空出两个汉字的距离,有的用户可能会使用“\hskip2\ccwd”命令,但由于10.3.2节中所指出的原因,这样留出的

空实际上多出一个西文的空格。该问题可使用 10.3.2 节所介绍的 `\ccnospace` 命令来解决, 即使用形如 `\ccnospace{\CS\hskip2\ccwd}` 的命令。更为简单的办法是输入两个区位码为 0101 的字符, 因为该字符不产生排版输出, 而它的宽度正好为一个汉字的宽度。

10.3.7 关于字符 “%” 的处理

字符 “%” 通常用来在源文件中引入说明。为了避免出现花括号不配对、汉字换行时产生多余的空格等麻烦, CCT 预处理程序在转换时不处理 “%” 后面的汉字, 而简单地将它们拷贝到输出文件中去。但如果 “%” 前面的字符是 “\”, 则 CCT 认为它不是说明字符而对跟随在它后面的汉字照常处理。

用户需要注意的是, 当排版正文中用到 “%” 时要注意写法以免引起错误。例如, 当使用 L^AT_EX 命令 “`\verb+%+`” 时, 排版源文件中同一行上字符 “%” 的后面不能有汉字, 否则将会出错。当然, 用户也可将命令写成其他形式, 如 “`\texttt{\%}`” 或 “`\texttt{\char37}`”。

10.3.8 数学公式中的中文

如果在数学环境中使用汉字, 则要用 `\hbox`, `\vbox` 或 `\mbox` 将汉字括起来, 如:

当 $x > 0$ 时 `$\mbox{ 当 }x>0\mbox{ 时}$`

10.4 CCT 预处理程序 CCT.EXE

CCT.EXE 将 CCT 源文件转换为 T_EX 源文件。用户在准备好 CCT 源文件后必须首先将其转换为 T_EX 源文件, 然后才能进行排版处理。

调用 CCT.EXE 的格式如下:

CCT [输入文件名 [输出文件名]]

其中输入文件名为用户的 CCT 源文件名, 输出文件名为转换得到的 T_EX 源文件名。如果省略输入文件名中的扩展名, 程序会自动加上 “.CTX” 作为扩展名; 如果省略输出文件名中的扩展名, 程序会自动加上 “.TEX” 作为扩展名; 如果省略输出文件名, 程序会取输入文件名加上扩展名 “.TEX” 作为输出文件名。例如命令:

```
cct test
```

等价于

```
cct test.ctx test.tex
```

CCT.EXE 支持下述命令行选项:

-M 产生一个与 T_EX 源文件同名, 但扩展名为 .MAP 的文件。这个文件中给出 CCT 源文件与 T_EX 源文件中的行号对应关系, 供用户在 CCT 源文件中查错时用。文件中的每行具有如下格式:

CCT 文件中的行号 ==> T_EX 文件中的行号

例如, 假设 “.MAP” 文件中包含下面一行:

5 ==> 6 7 8

则表示 CCT 源文件中的第 5 行对应着 $\text{T}_{\text{E}}\text{X}$ 文件中的 6, 7, 8 三行。

- S CCT.EXE 处理时将每个汉字码转换成对应的两个 ASCII 字符。如汉字“亮”被转换成两个“A”。如果转换后的码正好是 $\text{T}_{\text{E}}\text{X}$ 的保留字符，如“\”，“\$”等，则 CCT.EXE 将它们转换成“\char###”的形式，其中“###”代表字符的 ASCII 码，从而保证 $\text{T}_{\text{E}}\text{X}$ 能正确地处理这些字符。缺省情况下被转换成“\char###”形式的字符有下面一些：

\$ % & @ \ ^ _ { } ~

它们可满足大部分情况下的要求。但有时用户希望 CCT.EXE 将上述字符之外的某些字符也转换成“\char###”的形式，“-S”选项便是为此而设，它后面需跟随一个 ASCII 码（十进制数），表示将该字符也转换成“\char###”的形式。用户可以使用多个“-S”选项。例如，当使用“\index”命令及“MAKEIDX.EXE”程序生成索引时，字符“!”被用作二级索引命令，此时可使用“-S33”选项（33 是“!”的 ASCII 码）来使 CCT.EXE 将由汉字所产生的“!”写成“\char33”的形式，以避免“MAKEIDX.EXE”将其当做二级索引命令。

- H 显示一个简单的使用说明。

用户也可以在初始化文件 CCT.INI 中使用上述选项。

10.5 True Type 字体的使用

Windows 的 True Type 中文字体是随处可得的，而 CCT 中的中文字体则只有很少的几种。下面给出的步骤就可以在 CCT 中使用 Windows 的 True Type 字体。

如果你只想在打印 DVI 文件时可以利用 True Type 字体，那么需要进行如下操作：

1. 下载在 <http://202.38.68.78/~texguru/software/gbk.zip> 上的软件，并把它解开到 em $\text{T}_{\text{E}}\text{X}$ 目录中，覆盖原有的文件。然后执行 cctinit。
2. 执行 cctwin32 或者 cctwin16，打开菜单 File → Options...，在出现的对话框中把 Resolution 改为 300，并选择 Save option 框。按 OK。然后打开菜单 File → TrueType fonts ...，在新出现的对话框中，选择一种 \ziti{*}（比如说是 \ziti{M}），然后按 Define... 钮，出现 TrueType 字体框，从中选择你喜爱的字体。按 OK。然后可以继续选择 \ziti{*}，再进行定义。这样当你在 CCT 文档中利用了 \ziti{M} 命令，那么 cctwin32 或 cctwin16 就会以你刚才所选择的字体显示。

注意，这样从包含 \ziti{M} 的源文件生成的 .dvi 文件只能用 cctwin 查看，原来 DOS 下面的 dvisrc 则会抱怨找不到这种字体。

如果你想把所得的结果转化为 PS 文件，那么就进行如下操作：

1. 下载 cdiva.exe(<http://202.38.68.78/~texguru/software/Cdiva.exe>) 到 em $\text{T}_{\text{E}}\text{X}$ 目录中。
2. 执行 cdiva -c 命令，出现对话框。在这个对话框中，利用 --> 和 <-- 按钮翻到你所要定义的 \ziti{*}，然后在 TTF 字体表中选择你喜爱的字体，比如说对应于 \ziti{M}，我们选择了隶书字体。完成了一系列选择后，按确定钮，退出设置程序。

注意, 第一次执行这个程序时, 你会发现 PK 输出路径框是空的, 要在其中输入 `c:\emtex\pixel\` 或者 `d:\emtex\pixel\` 等字符串, 具体视 emT_EX 所处的位置改变盘符。

由于该对话框设计不完善, 因此不要随便使用 CCT 字体表旁的下拉箭头, 为了改变 `\ziti{}` 与 TrueType 字体的对应, 一定要利用 `-->` 和 `<--` 按钮。

3. 完成了上述设置, 在 CCT 文档中就可以使用 `\ziti{*}` 命令, 其参数就是你在设置程序中已定义了匹配的字母, 如 `\ziti{M}` (注意, 如果在设置程序中没有为 `\ziti{M}` 定义匹配的 TTF 字体, 在编译时不会显示错误信息, 但得不到所期望的转化结果)。
4. 制作如下的批处理文件:

```
@echo off
if exist %1.ctx call cct %1
call latex2e %1
call cdvia %1.dvi ~~tmp.dvi
dvidrv dvips32 ~~tmp.dvi
if exist %1.ps del %1.ps
ren ~~tmp.ps %1.ps
del ~~tmp.dvi
echo on
```

用其取代我们在第 226 页上给出的 `tp.bat` 程序。即可以完成从 DVI 到 PS 的完整转化, 其中字体用 `\ziti{M}` 的中文以隶书形式显示。

第四部分 附录

附录 A 符号汇总

我们在本章列出在正文中出现的各种符号，以方便使用时查找。虽然有些符号在正文中已详细讲解，我们仍把它们列在本章中。

A.1 正文符号

A.1.1 命令字符

$\$ = \backslash \$$ $\& = \backslash \&$ $\% = \backslash \%$ $\# = \backslash \#$ $_ = \backslash _$
 $\{ = \backslash \{$ $\} = \backslash \}$ $\backslash = \$\backslash \backslash \$$ $\^ = \backslash \^ \{ \}$ $_ = \backslash _ \{ \}$.

A.1.2 特殊字符

$\S = \backslash S$ $\dagger = \backslash dag$ $\ddagger = \backslash ddag$ $\P = \backslash P$ $\copyright = \backslash copyright$ $\pounds = \backslash pounds$

A.1.3 外文字母

$\oe = \backslash oe$ $\OE = \backslash OE$ $\ae = \backslash ae$ $\AE = \backslash AE$ $\aa = \backslash aa$ $\AA = \backslash AA$
 $\imath = \backslash i'$ $\oslash = \backslash o$ $\O = \backslash O$ $\ell = \backslash l$ $\L = \backslash L$ $\ss = \backslash ss$
 $\SS = \backslash SS$ $\imath = \backslash i'$

A.1.4 重音

$\grave{o} = \backslash ' \{ o \}$ $\acute{o} = \backslash ' \{ o \}$ $\hat{o} = \backslash ^ \{ o \}$ $\ddot{o} = \backslash " \{ o \}$ $\tilde{o} = \backslash \sim \{ o \}$
 $\bar{o} = \backslash = \{ o \}$ $\dot{o} = \backslash . \{ o \}$ $\ddot{o} = \backslash u \{ o \}$ $\ddot{o} = \backslash v \{ o \}$ $\ddot{o} = \backslash H \{ o \}$
 $\circ = \backslash t \{ oo \}$ $\grave{o} = \backslash c \{ o \}$ $\grave{o} = \backslash d \{ o \}$ $\grave{o} = \backslash b \{ o \}$ $\grave{o} = \backslash r \{ o \}$

(最后的命令 $\backslash r$ 是新出现在 \LaTeX 2_ϵ 中的。) 上面的 o 只是一个示例，实际上可以用任何字母。而对于字母 i 和 j ，在加上重音时必须去掉它们的点，为此就需要在这两个字母前面加上 \backslash 。命令 $\backslash i$ 和 $\backslash j$ 就会得到 i 和 j 。对于由非字母组成的重音命令，可以不使用大括号。

A.2 数学符号表

A.2.1 希伯来字母

输入	结果	输入	结果
<code>\aleph</code>	\aleph	<code>\beth</code>	\beth
<code>\daleth</code>	\daleth	<code>\gimel</code>	\gimel

除 `\aleph` 外所有字母需要 `amssymb` 宏包。

A.2.2 希腊字母

输入	结果	输入	结果	输入	结果	输入	结果
<code>\alpha</code>	α	<code>\beta</code>	β	<code>\gamma</code>	γ	<code>\digamma</code>	\digamma
<code>\delta</code>	δ	<code>\epsilon</code>	ϵ	<code>\varepsilon</code>	ε	<code>\zeta</code>	ζ
<code>\eta</code>	η	<code>\theta</code>	θ	<code>\vartheta</code>	ϑ	<code>\iota</code>	ι
<code>\kappa</code>	κ	<code>\varkappa</code>	\varkappa	<code>\lambda</code>	λ	<code>\mu</code>	μ
<code>\nu</code>	ν	<code>\xi</code>	ξ	<code>\pi</code>	π	<code>\varpi</code>	ϖ
<code>\rho</code>	ρ	<code>\varrho</code>	ϱ	<code>\sigma</code>	σ	<code>\varsigma</code>	ς
<code>\tau</code>	τ	<code>\upsilon</code>	υ	<code>\phi</code>	ϕ	<code>\varphi</code>	φ
<code>\chi</code>	χ	<code>\psi</code>	ψ	<code>\omega</code>	ω		

`\digamma` 和 `\varkappa` 需要 `amssymb` 宏包。

输入	结果	输入	结果	输入	结果	输入	结果
<code>\Gamma</code>	Γ	<code>\varGamma</code>	\varGamma	<code>\Delta</code>	Δ	<code>\varDelta</code>	\varDelta
<code>\Theta</code>	Θ	<code>\varTheta</code>	\varTheta	<code>\Lambda</code>	Λ	<code>\varLambda</code>	\varLambda
<code>\Xi</code>	Ξ	<code>\varXi</code>	\varXi	<code>\Pi</code>	Π	<code>\varPi</code>	\varPi
<code>\Sigma</code>	Σ	<code>\varSigma</code>	\varSigma	<code>\Upsilon</code>	Υ	<code>\varUpsilon</code>	\varUpsilon
<code>\Phi</code>	Φ	<code>\varPhi</code>	\varPhi	<code>\Omega</code>	Ω	<code>\varOmega</code>	\varOmega

名称以 `var` 开头的符号需要 `amsmath` 宏包。

A.2.3 L^AT_EX 的关系运算符

输入	结果	输入	结果	输入	结果	输入	结果
<code>\in</code>	\in	<code>\leq</code>	\leq	<code>\ll</code>	\ll	<code>\prec</code>	\prec
<code>\preceq</code>	\preceq	<code>\sim</code>	\sim	<code>\simeq</code>	\simeq	<code>\equiv</code>	\equiv
<code>\subset</code>	\subset	<code>\subseteq</code>	\subseteq	<code>\sqsubseteq</code>	\sqsubseteq	<code>\smile</code>	\smile
<code>\perp</code>	\perp	<code>\mid</code>	\mid	<code>\vdash</code>	\vdash	<code>\propto</code>	\propto
<code>\bowtie</code>	\bowtie	<code>\ni</code>	\ni	<code>\geq</code>	\geq	<code>\gg</code>	\gg
<code>\succ</code>	\succ	<code>\succeq</code>	\succeq	<code>\cong</code>	\cong	<code>\approx</code>	\approx
<code>\doteq</code>	\doteq	<code>\supset</code>	\supset	<code>\supseteq</code>	\supseteq	<code>\frown</code>	\frown
<code>\models</code>	\models	<code>\parallel</code>	\parallel	<code>\dashv</code>	\dashv	<code>\asymp</code>	\asymp
<code>\bowtie</code>	\bowtie						
<code>\sqsubset</code>	\sqsubset	<code>\sqsupset</code>	\sqsupset	<code>\Join</code>	\Join		

最后三个符号需要 `latexsym` 宏包。

输入	结果	输入	结果	输入	结果
<code>\leqslant</code>	\leqslant	<code>\geqslant</code>	\geqslant	<code>\eqslantless</code>	\leqslant
<code>\eqslantgtr</code>	\gtrless	<code>\lesssim</code>	\lesssim	<code>\gtrsim</code>	\gtrsim
<code>\lessapprox</code>	\lessapprox	<code>\gtrapprox</code>	\gtrapprox	<code>\approxeq</code>	\approx
<code>\lessdot</code>	\lessdot	<code>\gtrdot</code>	\gtrdot	<code>\lll</code>	\lll
<code>\ggg</code>	\ggg	<code>\lessgtr</code>	\lessgtr	<code>\gtrless</code>	\gtrless
<code>\lesseqgtr</code>	\lesseqgtr	<code>\gtreqless</code>	\gtreqless	<code>\lesseqqgtr</code>	\lesseqqgtr
<code>\gtreqqless</code>	\gtreqqless	<code>\doteqdot</code>	\doteqdot	<code>\eqcirc</code>	\circ
<code>\circeq</code>	\circ	<code>\fallingdotseq</code>	\fallingdotseq	<code>\risingdotseq</code>	\risingdotseq
<code>\triangleq</code>	\triangleq	<code>\backsim</code>	\sim	<code>\thicksim</code>	\sim
<code>\backsimeq</code>	\backsimeq	<code>\thickapprox</code>	\approx	<code>\preccurlyeq</code>	\preccurlyeq
<code>\succcurlyeq</code>	\succcurlyeq	<code>\curlyeqprec</code>	\curlyeqprec	<code>\curlyeqsucc</code>	\curlyeqsucc
<code>\precsim</code>	\precsim	<code>\succsim</code>	\succsim	<code>\precapprox</code>	\precapprox
<code>\succapprox</code>	\succapprox	<code>\subteqq</code>	\subsetneqq	<code>\supseteqq</code>	\supseteqq
<code>\Subset</code>	\Subset	<code>\Supset</code>	\Supset	<code>\vartriangleleft</code>	\triangleleft
<code>\vartriangleright</code>	\vartriangleright	<code>\trianglelefteq</code>	\trianglelefteq	<code>\trianglerighteq</code>	\trianglerighteq
<code>\vDash</code>	\vDash	<code>\Vdash</code>	\Vdash	<code>\Vvdash</code>	\Vdash
<code>\smallsmile</code>	\smile	<code>\smallfrown</code>	\frown	<code>\shortmid</code>	\mid
<code>\shortparallel</code>	\parallel	<code>\bumpeq</code>	\bumpeq	<code>\Bumpeq</code>	\Bumpeq
<code>\between</code>	\between	<code>\pitchfork</code>	\pitchfork	<code>\varpropto</code>	\propto
<code>\backepsilon</code>	ϵ	<code>\blacktriangleleft</code>	\blacktriangleleft	<code>\blacktriangleright</code>	\blacktriangleright
<code>\therefore</code>	\therefore	<code>\because</code>	\because		

上述运算符都需要 `amssymb` 宏包。

输入	结果	输入	结果	输入	结果
<code>\neq</code>	\neq	<code>\notin</code>	\notin	<code>\nless</code>	\nless
<code>\ngtr</code>	\ngtr	<code>\nleq</code>	\nleq	<code>\ngeq</code>	\ngeq
<code>\nleqslant</code>	\nleqslant	<code>\ngeqslant</code>	\ngeqslant	<code>\nleqq</code>	\nleqq
<code>\ngeqq</code>	\ngeqq	<code>\lneq</code>	\lneq	<code>\gneq</code>	\gneq
<code>\lneqq</code>	\lneqq	<code>\gneqq</code>	\gneqq	<code>\lvertneqq</code>	\lvertneqq
<code>\gvertneqq</code>	\gvertneqq	<code>\lnsim</code>	\lnsim	<code>\gnsim</code>	\gnsim
<code>\lnapprox</code>	\lnapprox	<code>\gnapprox</code>	\gnapprox	<code>\nprec</code>	\nprec
<code>\npreceq</code>	\npreceq	<code>\nsucc</code>	\nsucc	<code>\npreceq</code>	\npreceq
<code>\nsucceq</code>	\nsucceq	<code>\precneqq</code>	\precneqq	<code>\succnsim</code>	\succnsim
<code>\precnapprox</code>	\precnapprox	<code>\succnapprox</code>	\succnapprox	<code>\nsim</code>	\nsim
<code>\ncong</code>	\ncong	<code>\nshortmid</code>	\nshortmid	<code>\nshortparallel</code>	\nshortparallel
<code>\nmid</code>	\nmid	<code>\nparallel</code>	\nparallel	<code>\nvdash</code>	\nvdash

输入	结果	输入	结果	输入	结果
<code>\nvDash</code>	\nVdash	<code>\nVdash</code>	\nVdash	<code>\nVDash</code>	\nVDash
<code>\ntriangleleft</code>	\ntriangleleft	<code>\ntriangleright</code>	\ntriangleright	<code>\ntrianglelefteq</code>	\ntrianglelefteq
<code>\ntrianglerighteq</code>	\ntrianglerighteq	<code>\nsubseteq</code>	\nsubseteq	<code>\nsupseteq</code>	\nsupseteq
<code>\nsubseteqq</code>	\nsubseteqq	<code>\nsupseteqq</code>	\nsupseteqq	<code>\subsetneq</code>	\subsetneq
<code>\supsetneq</code>	\supsetneq	<code>\varsubsetneq</code>	\varsubsetneq	<code>\varsupsetneq</code>	\varsupsetneq
<code>\subsetneqq</code>	\subsetneqq	<code>\supsetneqq</code>	\supsetneqq	<code>\varsubsetneqq</code>	\varsubsetneqq
<code>\varsupsetneqq</code>	\varsupsetneqq				

除 `\ne` 和 `\notin` 外的其他符号都需要 `amssymb` 宏包。

A.2.4 二元运算符

输入	结果	输入	结果	输入	结果
<code>\pm</code>	\pm	<code>\mp</code>	\mp	<code>\times</code>	\times
<code>\cdot</code>	\cdot	<code>\circ</code>	\circ	<code>\bigcirc</code>	\bigcirc
<code>\div</code>	\div	<code>\diamond</code>	\diamond	<code>\ast</code>	\ast
<code>\star</code>	\star	<code>\cap</code>	\cap	<code>\cup</code>	\cup
<code>\sqcap</code>	\sqcap	<code>\sqcup</code>	\sqcup	<code>\wedge</code>	\wedge
<code>\triangleleft</code>	\triangleleft	<code>\triangleright</code>	\triangleright	<code>\bigtriangleup</code>	\bigtriangleup
<code>\bigtriangledown</code>	\bigtriangledown	<code>\oplus</code>	\oplus	<code>\ominus</code>	\ominus
<code>\otimes</code>	\otimes	<code>\oslash</code>	\oslash	<code>\odot</code>	\odot
<code>\bullet</code>	\bullet	<code>\dagger</code>	\dagger	<code>\ddagger</code>	\ddagger
<code>\setminus</code>	\setminus	<code>\uplus</code>	\uplus	<code>\wr</code>	\wr
<code>\amalg</code>	\amalg	<code>\vee</code>	\vee		
<code>\lhd</code>	\lhd	<code>\rhd</code>	\rhd	<code>\unlhd</code>	\unlhd
<code>\unrhd</code>	\unrhd				

输入	结果	输入	结果	输入	结果
<code>\dotplus</code>	\dotplus	<code>\centerdot</code>	\centerdot	<code>\ltimes</code>	\ltimes
<code>\rtimes</code>	\rtimes	<code>\leftthreetimes</code>	\leftthreetimes	<code>\rightthreetimes</code>	\rightthreetimes
<code>\circleddash</code>	\circleddash	<code>\smallsetminus</code>	\smallsetminus	<code>\barwedge</code>	\barwedge
<code>\doublebarwedge</code>	\doublebarwedge	<code>\curlywedge</code>	\curlywedge	<code>\curlyvee</code>	\curlyvee
<code>\veebar</code>	\veebar	<code>\intercal</code>	\intercal	<code>\Cap</code>	\Cap
<code>\Cup</code>	\Cup	<code>\circledast</code>	\circledast	<code>\circledcirc</code>	\circledcirc
<code>\boxminus</code>	\boxminus	<code>\boxtimes</code>	\boxtimes	<code>\boxdot</code>	\boxdot
<code>\boxplus</code>	\boxplus	<code>\divideontimes</code>	\divideontimes		
<code>\And</code>	$\&$				

这四部分二元运算符，第一部分是标准 L^AT_EX 所提供的，第二部分需要 `latexsym` 宏包，第三部分是 `amssymb` 提供的，而最后一部分是在 `amsmath` 中。

A.2.5 箭头

输入	结果	输入	结果	输入	结果
<code>\leftarrow</code>	\leftarrow	<code>\rightarrow</code>	\rightarrow	<code>\longleftarrow</code>	\longleftarrow
<code>\longrightarrow</code>	\longrightarrow	<code>\Leftarrow</code>	\Leftarrow	<code>\Rightarrow</code>	\Rightarrow
<code>\Longleftarrow</code>	\Longleftarrow	<code>\Longrightarrow</code>	\Longrightarrow	<code>\leftrightarrow</code>	\leftrightarrow
<code>\Uparrow</code>	\Uparrow	<code>\Leftrightarrow</code>	\Leftrightarrow	<code>\Longleftrightarrow</code>	\Longleftrightarrow
<code>\uparrow</code>	\uparrow	<code>\downarrow</code>	\downarrow	<code>\longleftrightarrow</code>	\longleftrightarrow
<code>\Downarrow</code>	\Downarrow	<code>\updownarrow</code>	\updownarrow	<code>\Updownarrow</code>	\Updownarrow
<code>\nearrow</code>	\nearrow	<code>\searrow</code>	\searrow	<code>\swarrow</code>	\swarrow
<code>\nwarrow</code>	\nwarrow	<code>\mapsto</code>	\mapsto	<code>\longmapsto</code>	\longmapsto
<code>\hookrightarrow</code>	\hookrightarrow	<code>\hookleftarrow</code>	\hookleftarrow	<code>\rightharpoonup</code>	\rightharpoonup
<code>\leftharpoonup</code>	\leftharpoonup	<code>\leftharpoonup</code>	\leftharpoonup	<code>\rightleftharpoons</code>	\rightleftharpoons
<code>\leadsto</code>	\leadsto				
<code>\leftleftarrows</code>	\leftleftarrows	<code>\rightrightarrows</code>	\rightrightarrows	<code>\leftrightarrows</code>	\leftrightarrows
<code>\rightleftarrows</code>	\rightleftarrows	<code>\Lleftarrow</code>	\Lleftarrow	<code>\Rrightarrow</code>	\Rrightarrow
<code>\twoheadleftarrow</code>	\twoheadleftarrow	<code>\twoheadrightarrow</code>	\twoheadrightarrow	<code>\leftarrowtail</code>	\leftarrowtail
<code>\rightarrowtail</code>	\rightarrowtail	<code>\looparrowleft</code>	\looparrowleft	<code>\looparrowright</code>	\looparrowright
<code>\upuparrows</code>	\upuparrows	<code>\downdownarrows</code>	\downdownarrows	<code>\upharpoonleft</code>	\upharpoonleft
<code>\upharpoonright</code>	\upharpoonright	<code>\downharpoonleft</code>	\downharpoonleft	<code>\downharpoonright</code>	\downharpoonright
<code>\rightsquigarrow</code>	\rightsquigarrow	<code>\multimap</code>	\multimap	<code>\leftrightsquigarrow</code>	\leftrightsquigarrow
<code>\nleftarrow</code>	\nleftarrow	<code>\nrightarrow</code>	\nrightarrow	<code>\nLeftarrow</code>	\nLeftarrow
<code>\nrightarrow</code>	\nrightarrow	<code>\nleftrightarrow</code>	\nleftrightarrow	<code>\nLeftrightarrow</code>	\nLeftrightarrow

表格中第一部分为标准 L^AT_EX 符号，第二部分只有一个符号，它需要 latexsym 宏包，第三部分是 A_MS 的符号，全部需要 amssymb 宏包，最后一部分是否定的箭头符号，也要使用 amssymb 宏包。其中 `\Longleftrightarrow` 也可以用 `\iff` 生成，只是后者要比前者在两端多些空白。

A.2.6 杂类符号

输入	结果	输入	结果	输入	结果
<code>\hbar</code>	\hbar	<code>\ell</code>	ℓ	<code>\imath</code>	\imath
<code>\jmath</code>	\jmath	<code>\wp</code>	\wp	<code>\Re</code>	\Re
<code>\Im</code>	\Im	<code>\partial</code>	∂	<code>\infty</code>	∞
<code>\prime</code>	\prime	<code>\emptyset</code>	\emptyset	<code>\backslash</code>	\backslash
<code>\forall</code>	\forall	<code>\exists</code>	\exists	<code>\smallint</code>	\smallint
<code>\triangle</code>	\triangle	<code>\surd</code>	\surd	<code>\Vert</code>	\Vert
<code>\top</code>	\top	<code>\bot</code>	\bot	<code>\P</code>	\P

输入	结果	输入	结果	输入	结果
<code>\S</code>	\S	<code>\dag</code>	\dagger	<code>\ddag</code>	\ddagger
<code>\flat</code>	\flat	<code>\natural</code>	\natural	<code>\sharp</code>	\sharp
<code>\angle</code>	\angle	<code>\clubsuit</code>	\clubsuit	<code>\diamondsuit</code>	\diamondsuit
<code>\heartsuit</code>	\heartsuit	<code>\spadesuit</code>	\spadesuit	<code>\neg</code>	\neg
<code>\Box</code>	\Box	<code>\Diamond</code>	\Diamond	<code>\mho</code>	\mho
<code>\hslash</code>	\hslash	<code>\complement</code>	\complement	<code>\backprime</code>	\backprime
<code>\vartriangle</code>	\triangle	<code>\Bbbk</code>	\mathbb{k}	<code>\varnothing</code>	\varnothing
<code>\diagup</code>	\diagup	<code>\diagdown</code>	\diagdown	<code>\blacktriangle</code>	\blacktriangle
<code>\blacktriangledown</code>	\blacktriangledown	<code>\triangledown</code>	\triangledown	<code>\Game</code>	\Game
<code>\square</code>	\square	<code>\blacksquare</code>	\blacksquare	<code>\lozenge</code>	\lozenge
<code>\blacklozenge</code>	\blacklozenge	<code>\measuredangle</code>	\measuredangle	<code>\sphericalangle</code>	\sphericalangle
<code>\circledS</code>	\circledS	<code>\bigstar</code>	\bigstar	<code>\Finv</code>	\Finv
<code>\eth</code>	\eth	<code>\nexists</code>	\nexists		

第一部分是标准的 L^AT_EX 符号, 第二部分需要 latexsym 宏包, 最后一部分符号来自于 \mathcal{A} M_S, 需要 amssymb 宏包。

A.2.7 数学重音

输入	结果	输入	结果	输入	结果	输入	结果
<code>\hat{a}</code>	\hat{a}	<code>\tilde{a}</code>	\tilde{a}	<code>\acute{a}</code>	\acute{a}	<code>\bar{a}</code>	\bar{a}
<code>\breve{a}</code>	\breve{a}	<code>\check{a}</code>	\check{a}	<code>\dot{a}</code>	\dot{a}	<code>\ddot{a}</code>	\ddot{a}
<code>\grave{a}</code>	\grave{a}	<code>\vec{a}</code>	\vec{a}	<code>\widehat{a}</code>	\widehat{a}	<code>\widetilde{a}</code>	\widetilde{a}
<code>\dddota</code>	\dddot{a}	<code>\ddddota</code>	\ddddot{a}				

上述重音都是标准 L^AT_EX 所提供的。另外需要注意的是, 如果给字母 i 和 j 加重音, 应该用 `\imath` 和 `\jmath`, 例如 `\hat{\imath}` 的结果为 \hat{i} 。

A.2.8 函数名称

<code>\arccos</code>	\arccos	<code>\arcsin</code>	\arcsin	<code>\arctan</code>	\arctan	<code>\arg</code>	\arg
<code>\cos</code>	\cos	<code>\cosh</code>	\cosh	<code>\cot</code>	\cot	<code>\coth</code>	\coth
<code>\csc</code>	\csc	<code>\deg</code>	\deg	<code>\det</code>	\det	<code>\dim</code>	\dim
<code>\exp</code>	\exp	<code>\gcd</code>	\gcd	<code>\hom</code>	\hom	<code>\inf</code>	\inf
<code>\injlim</code>	injlim	<code>\ker</code>	\ker	<code>\lim</code>	\lim	<code>\liminf</code>	\liminf
<code>\limsup</code>	\limsup	<code>\lg</code>	\lg	<code>\ln</code>	\ln	<code>\log</code>	\log
<code>\max</code>	\max	<code>\min</code>	\min	<code>\projlim</code>	$\operatorname{projlim}$	<code>\Pr</code>	\Pr
<code>\sec</code>	\sec	<code>\sin</code>	\sin	<code>\sinh</code>	\sinh	<code>\sup</code>	\sup
<code>\tan</code>	\tan	<code>\tanh</code>	\tanh	<code>\varliminf</code>	\varliminf	<code>\varlimsup</code>	\varlimsup
<code>\varinjlim</code>	\varinjlim	<code>\varprojlim</code>	\varprojlim				

A.2.9 具有两种尺寸的运算符

输入	正文公式	单独公式	输入	正文公式	单独公式
<code>\prod_{i=1}^n</code>	$\prod_{i=1}^n$	$\prod_{i=1}^n$	<code>\coprod_{i=1}^n</code>	$\coprod_{i=1}^n$	$\coprod_{i=1}^n$
<code>\bigcap_{i=1}^n</code>	$\bigcap_{i=1}^n$	$\bigcap_{i=1}^n$	<code>\bigcup_{i=1}^n</code>	$\bigcup_{i=1}^n$	$\bigcup_{i=1}^n$
<code>\bigwedge_{i=1}^n</code>	$\bigwedge_{i=1}^n$	$\bigwedge_{i=1}^n$	<code>\bigvee_{i=1}^n</code>	$\bigvee_{i=1}^n$	$\bigvee_{i=1}^n$
<code>\bigsqcup_{i=1}^n</code>	$\bigsqcup_{i=1}^n$	$\bigsqcup_{i=1}^n$	<code>\biguplus_{i=1}^n</code>	$\biguplus_{i=1}^n$	$\biguplus_{i=1}^n$
<code>\bigotimes_{i=1}^n</code>	$\bigotimes_{i=1}^n$	$\bigotimes_{i=1}^n$	<code>\bigoplus_{i=1}^n</code>	$\bigoplus_{i=1}^n$	$\bigoplus_{i=1}^n$
<code>\bigodot_{i=1}^n</code>	$\bigodot_{i=1}^n$	$\bigodot_{i=1}^n$	<code>\sum_{i=1}^n</code>	$\sum_{i=1}^n$	$\sum_{i=1}^n$
<code>\int_0^1</code>	\int_0^1	\int_0^1	<code>\oint_0^1</code>	\oint_0^1	\oint_0^1

在单独公式中，如果不想把上下标做为上下限对待，可以把 `\nolimits` 命令接在符号名称的后面。另一方面，如果在正文公式中想把上下标做为上下限对待，就在符号名称后面接 `\limits` 命令。对于其他积分符号，见 5.2.5 节中的介绍。

A.3 字符表

除 `cminch` 外所有标准 $\text{T}_{\text{E}}\text{X}$ 字体都由 128 个字符组成，在 $\text{T}_{\text{E}}\text{X}$ 内部给它们赋为 0 到 127 的数值。在字体中字符的布局见下面的表格。这里用的是八进制，而相应的十进制数放在对应符号旁边。

字体布局 1 表示的是罗马字体 `cmr10` 的字符对应关系，这是真正的 OT1 编码框架，是绝大多数具有同样符号的字体代表，因为可能不同的样式中，这些符号具有相同的位置。文本斜体只是稍微有点儿区别。而大写及小写字体 `cmcsc10` 则偏离得就有点几多了，因为这时没有了连写。打字机字体则与布局 1 中的 OT1 标准顺序差得更远了。最后，数学和符号字体则具有截然不同的对应关系，因为其内容与文本字体并没有任何关联。所有这些有偏差的字体模式演示在布局 2-8 中。

注意：将来 $\text{T}_{\text{E}}\text{X}3.0$ 的字体中应包含 256 个字符。它们要遵从 T1 对应关系。

A.3.1 `cmr10` 字体

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Υ 7
'01x	Φ 8	Ψ 9	Ω 10	ff 11	fi 12	fl 13	ffi 14	ffl 15
'02x	ı 16	ı 17	` 18	' 19	˘ 20	˙ 21	˚ 22	° 23
'03x	, 24	ß 25	æ 26	œ 27	ø 28	Æ 29	Œ 30	Ø 31

	0	1	2	3	4	5	6	7
'04x	- 32	! 33	" 34	# 35	\$ 36	% 37	& 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	ı 60	= 61	ı 62	? 63
'10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	[91	" 92] 93	^ 94	˘ 95
'14x	‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	- 123	— 124	" 125	~ 126	¨ 127

字体布局 1 字符字体为 `cmr10`。这里给出的是字符与相应数值的标准对应关系。在下面的布局中给出的是非标准赋值。

A.3.2 cmcsc10 字体

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Υ 7
'01x	Φ 8	Ψ 9	Ω 10	↑ 11	↓ 12	' 13	ı 14	ı 15
'02x	ı 16	J 17	` 18	˘ 19	˘ 20	˘ 21	˘ 22	˘ 23
'03x	, 24	ss 25	Æ 26	œ 27	ø 28	Æ 29	œ 30	Ø 31
'04x	- 32	! 33	" 34	# 35	\$ 36	% 37	& 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
'10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	[91	" 92] 93	^ 94	˘ 95
'14x	‘ 96	A 97	B 98	C 99	D 100	E 101	F 102	G 103
'15x	H 104	I 105	J 106	K 107	L 108	M 109	N 110	O 111
'16x	P 112	Q 113	R 114	S 115	T 116	U 117	V 118	W 119
'17x	X 120	Y 121	Z 122	- 123	— 124	" 125	~ 126	¨ 127

字体布局 2 字符字体为 `cmcsc10`。与布局 1 的差别在于 11-15, 60 和 62 号符号。通常位于 11-15 号的连写被其他一些符号所取代。

A.3.3 cmti10 字体

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	Α 3	Ξ 4	Π 5	Σ 6	Υ 7
'01x	Φ 8	Ψ 9	Ω 10	ff 11	fī 12	fl 13	ffi 14	ffl 15
'02x	ı 16	ı̇ 17	ˆ 18	ˆ 19	ˆ 20	ˆ 21	ˆ 22	ˆ 23
'03x	ı̇ 24	β 25	æ 26	æ 27	ø 28	Æ 29	Œ 30	Ø 31
'04x	ˆ 32	! 33	" 34	# 35	£ 36	% 37	€ 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	i 60	= 61	ı̇ 62	? 63
'10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	/ 91	" 92	/ 93	^ 94	' 95
'14x	' 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	- 123	— 124	" 125	~ 126	ˆ 127

字体布局 3 字符字体为 cmti10。与布局 1 的差别在于第 36 号 (£ 代替了 \$) 和第 38 号 (€ 代替了 &) 符号。所有其他文本斜体都具有与之相同的模式。

A.3.4 cmitt10 字体

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	Α 3	Ξ 4	Π 5	Σ 6	Τ 7
'01x	Φ 8	Ψ 9	Ω 10	↑ 11	↓ 12	' 13	ı̇ 14	ı̇ 15
'02x	ı̇ 16	ı̇ 17	ˆ 18	ˆ 19	ˆ 20	ˆ 21	ˆ 22	ˆ 23
'03x	ı̇ 24	β 25	æ 26	æ 27	ø 28	Æ 29	Œ 30	Ø 31
'04x	ı̇ 32	! 33	" 34	# 35	\$ 36	% 37	& 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
'10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	[91	\ 92] 93	^ 94	_ 95
'14x	' 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	{ 123	124	} 125	~ 126	ˆ 127

字体布局 4 字符字体为 cmitt10。所有的 tt 字体都具有同样的模式，只是 cmvtt10 例外，其模式与布局 1 一致。差别在于 11-15, 60, 62, 92, 123, 124 号。而且斜体打字

机字体 cmtit 用斜体字体 ε 和 ϑ 取代了第 36、38 号符号。cmtcsc10 字体在 97-122 号是更小的大写字母。

A.3.5 cmtex10 字体

	0	1	2	3	4	5	6	7
'00x	· 0	↓ 1	α 2	β 3	Λ 4	\neg 5	ϵ 6	π 7
'01x	λ 8	γ 9	δ 10	\uparrow 11	\pm 12	\otimes 13	\wp 14	ϑ 15
'02x	\subset 16	\supset 17	\cap 18	\cup 19	\forall 20	\exists 21	\otimes 22	\S 23
'03x	\leftarrow 24	\rightarrow 25	\neq 26	\diamond 27	\leq 28	\geq 29	\equiv 30	\vee 31
'04x	32	! 33	" 34	# 35	\$ 36	% 37	& 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
'10x	Q 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	[91	\ 92] 93	^ 94	_ 95
'14x	' 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	{ 123	124	} 125	~ 126	f 127

字体布局 5 字符字体为 cmtex10。与布局 4 的差别在于 0-31 号和 127 号字符，现在这里包含的是特殊数学符号。

A.3.6 cmml10 字体

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	A 3	Ξ 4	Π 5	Σ 6	Υ 7
'01x	Φ 8	Ψ 9	Ω 10	α 11	β 12	γ 13	δ 14	ϵ 15
'02x	ζ 16	η 17	θ 18	ι 19	κ 20	λ 21	μ 22	ν 23
'03x	ξ 24	π 25	ρ 26	σ 27	τ 28	υ 29	ϕ 30	χ 31
'04x	ψ 32	ω 33	ε 34	ϑ 35	ϖ 36	ϱ 37	ς 38	φ 39
'05x	\leftarrow 40	\leftarrow 41	\rightarrow 42	\rightarrow 43	\curvearrowright 44	\curvearrowleft 45	\triangleright 46	\triangleleft 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	. 58	, 59	< 60	/ 61	> 62	* 63
'10x	ϑ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	\flat 91	\natural 92	\sharp 93	\smile 94	\frown 95
'14x	ℓ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	i 123	j 124	φ 125	\sim 126	\sim 127

字体布局 6 字符字体为 `cmmi10`。包括 `cmmb10` 在内的 `cmmi` 族字体, 11-39 号上是小写希腊字母, 40-47, 60 64 和 123 127 号上是一些数学符号。

A.3.7 `cmsy10` 字体

	0	1	2	3	4	5	6	7
'00x	— 0	· 1	× 2	* 3	÷ 4	◇ 5	± 6	∓ 7
'01x	⊕ 8	⊖ 9	⊗ 10	⊙ 11	⊙ 12	○ 13	◦ 14	● 15
'02x	≍ 16	≡ 17	⊆ 18	⊇ 19	≤ 20	≥ 21	≤ 22	≥ 23
'03x	~ 24	≈ 25	⊂ 26	⊃ 27	≪ 28	≫ 29	⋈ 30	⋈ 31
'04x	← 32	→ 33	↑ 34	↓ 35	↔ 36	↗ 37	↘ 38	≈ 39
'05x	⇐ 40	⇒ 41	⇑ 42	⇓ 43	⇔ 44	↖ 45	↙ 46	∝ 47
'06x	/ 48	∞ 49	∈ 50	∋ 51	△ 52	▽ 53	/ 54	· 55
'07x	√ 56	∃ 57	¬ 58	∅ 59	℔ 60	℔ 61	⊤ 62	⊥ 63
'10x	ℕ 64	ℒ 65	℔ 66	℔ 67	℔ 68	℔ 69	℔ 70	℔ 71
'11x	℔ 72	℔ 73	℔ 74	℔ 75	℔ 76	℔ 77	℔ 78	℔ 79
'12x	℔ 80	℔ 81	℔ 82	℔ 83	℔ 84	℔ 85	℔ 86	℔ 87
'13x	℔ 88	℔ 89	℔ 90	℔ 91	℔ 92	℔ 93	℔ 94	℔ 95
'14x	⊢ 96	⊣ 97	⊤ 98	⊥ 99	⌈ 100	⌋ 101	{ 102	} 103
'15x	< 104	> 105	106	107	↑ 108	↓ 109	\ 110	∩ 111
'16x	√ 112	∏ 113	∇ 114	∫ 115	⊠ 116	⊡ 117	⊢ 118	⊣ 119
'17x	§ 120	† 121	‡ 122	¶ 123	♣ 124	◇ 125	♥ 126	♣ 127

字体布局 7 字符字体为 `cmsy10`。在 `cmsy` 字体中有许多数学符号, 以及 65-90 号位置上是花体字母 *A...Z*。黑体版本 `cmbsy10` 具有完全一样的模式。

A.3.8 `cmex10` 字体

	0	1	2	3	4	5	6	7
'00x	(0)	[1]	[2]	[3]	[4]	[5]	[6]	[7]
'01x	{ 8 }	{ 9 }	< 10	> 11	12	13	/ 14	\ 15
'02x	(16)	(17)	(18)	(19)	[20]	[21]	[22]	[23]
'03x	[24]	[25]	{ 26 }	{ 27 }	< 28	> 29	/ 30	\ 31
'04x	(32)	(33)	[34]	[35]	[36]	[37]	[38]	[39]
'05x	{ 40 }	{ 41 }	< 42	> 43	/ 44	\ 45	/ 46	\ 47
'06x	(48)	(49)	[50]	[51]	[52]	[53]	54	55

	0	1	2	3	4	5	6	7
'07x	56	57	58	59	60	61	62	63
'10x	64	65	66	67	68	69	70	71
'11x	72	73	74	75	76	77	78	79
'12x	80	81	82	83	84	85	86	87
'13x	88	89	90	91	92	93	94	95
'14x	96	97	98	99	100	101	102	103
'15x	104	105	106	107	108	109	110	111
'16x	112	113	114	115	116	117	118	119
'17x	120	121	122	123	124	125	126	127

字体布局 8 字符字体为 cmex10，由外观尺寸可以变化的各种数学符号组成。

A.3.9 wncyr10 字体

	0	1	2	3	4	5	6	7
'00x	Ѓ 0	Ѕ 1	Ї 2	Э 3	І 4	Є 5	Ђ 6	Ѓ 7
'01x	Ѕ 8	Ѕ 9	ц 10	э 11	і 12	є 13	ђ 14	ћ 15
'02x	Ю 16	Ж 17	Й 18	Ё 19	У 20	Ө 21	Ѕ 22	Я 23
'03x	ю 24	ж 25	й 26	ё 27	у 28	ө 29	ѕ 30	я 31
'04x	“ 32	! 33	” 34	Ђ 35	~ 36	% 37	’ 38	’ 39
'05x	(40) 41	* 42	Ђ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	« 60	ı 61	» 62	? 63
'10x	˘ 64	А 65	Б 66	Ц 67	Д 68	Е 69	Ф 70	Г 71
'11x	Х 72	И 73	Ј 74	К 75	Л 76	М 77	Н 78	О 79
'12x	П 80	Ч 81	Р 82	С 83	Т 84	У 85	В 86	Ш 87
'13x	Ш 88	Ы 89	З 90	[91	“ 92] 93	Ь 94	Ъ 95
'14x	‘ 96	а 97	б 98	ц 99	д 100	е 101	ф 102	г 103
'15x	х 104	и 105	ј 106	к 107	л 108	м 109	н 110	о 111
'16x	п 112	ч 113	р 114	с 115	т 116	у 117	в 118	ш 119
'17x	ш 120	ы 121	з 122	– 123	— 124	№ 125	ь 126	ъ 127

字体布局 9 字符字体为 wncyr10，这是一种由华盛顿大学提供的 Cyrillic 字体。

附录 B 参考文献数据库的处理

在科技出版物的创作中，参考文献的组织是一个不可缺少的标准过程。我们已经在 4.3.6 节和 8.3.2 节中讲解了如何利用 thebibliography 环境来排参考文献，以及如何在正文中引用其中的条目。有时候作者会发现，在绝大多数文章中，他经常引用同样的作品。与此类似，在同一领域工作的研究人员参考的论文也大致相同。这也就是说，在不同文档的 thebibliography 环境中经常有大量重复的条目，在一个研究所中的同事与同事之间处理的文档里也有类似的情形。

如果我们能把参考文献条目全部存贮在一个数据库文件里，使得所有的文档都可以引用其中的一组文献，就会节省大量的时间。借助于 L^AT_EX 提供的软件 BibT_EX，我们就可以建立起这样的一个数据库：把各种出版物的信息存贮在一个或多个后缀为 .bib 的文件中。每篇文献有一个关键词来与其他文献区分开，在正文中可以通过它来引用该文献。这样的文件就称为参考文献数据库。

B.1 BibT_EX 程序

BibT_EX 是 L^AT_EX 的一个辅助程序，它通过搜索一个或多个数据库，自动为 L^AT_EX 文档构造参考文献。要做到这一点，L^AT_EX 文件中必须在参考文献所位于的地方调用命令

```
\bibliography{数据库一,数据库二,...}
```

其中的参数值 数据库一, 数据库二, ... 就是要搜索的参考文献数据库的基本名，中间用逗号分开。并不需要显式地给出 .bib 后缀。

在 L^AT_EX 正文的任意地方都可以通过下面的命令引用数据库中的一篇文献：

```
\cite{关键词}
```

这与 8.3.2 节中的说明一样。其中 关键词 就是文献的标志，这当然要求用户必须预先知道这一信息。在第一次运行 L^AT_EX 后，就必须执行 BibT_EX 程序。至于如何调用这个程序，那就要看所处的计算机操作系统了，但通常的方法就是操作命令 bibtex 后接 L^AT_EX 文件的基本名。假设这个基本名是 comets，那么

```
bibtex comets
```

就会生成一个名为 comets.bbl 的新文件，其中就包含从数据库中提取的相应于 \cite 引用的参考文献信息，并包装到一个 thebibliography 环境中，下次运行 L^AT_EX 时就会把它输入到文档中。

有时候需要包含进一个在正文中没有引用的文献。这可以用如下命令来加入该文献：

```
\nocite{关键词}
```

可以把这条命令放在主文档内的任意地方。它并不生成任何文本，只是告诉 BibT_EX，也要把这一项放到参考文献中。而命令 \nocite{*} 会把数据库中所有项都包含到文档中，

这对于要生成所有条目及其关键词是非常有用的。（注意：对 BibTeX 0.98 版及以前版本这条命令是没有用的。）

在执行 BibTeX 后，就会生成 .bbl 文件，这就需要至少执行两次 L^AT_EX，以建立起参考文献和正文中正确的引用记号。参考文献就显示在 \bibliography 命令被调用的地方；这条命令实际上是把 .bbl 文件包含到正文中。

可以用下面的声明来选择参考文献的样式：

\bibliographystyle{样式}

可以在导言的任何地方调用这条命令。样式参数可以取如下的几个值：

- plain 参考文献中的条目按字母顺序排列；每项条目都有一个活动编号，并放在方括号内，这就是在正文中引用该文献（即 \cite 命令调用的地方）时的标记。
- unsorted 条目按第一次出现在 \cite 和 \nocite 命令中的顺序排列。在第一个 \cite 中的条目编号为 1，下一个具有不同关键词的 \cite 条目编号为 2，依此类推。其他标记和列表方式与 plain 样式相同。
- alpha 参考文献的顺序与 plain 样式相同，但是记号用的是作者姓名的缩写加上出版年代。对 Smith(1987) 的引用将会是 [Smi87]。
- abbrev 顺序和标记与 plain 相同，但是参考文献中要对作者的名字、月份、以及杂志名称进行缩写，以缩短长度。

在用户使用的计算机上可能还有其他可用的样式。特别地，正如 B.3.1 节所描述的作者-年代索引，我们可以从标准 L^AT_EX 宏包中衍生出大量的不同样式。参考文献样式包含在后缀为 .bst 的文件中。

下一节就讲解供 BibTeX 程序 0.99 版本使用的参考文献数据库。在这个版本中有很多 0.98 版本所没有的功能，我们用双星号 ** 来标明这些新功能。那些在旧版本中可以使用的数据库文件 (.bib 文件) 在 0.99 版本中仍具有相同的功能。然而，适用于一个版本的 .bst 参考文献样式文件在另一个版本中可能就行不通了。要想知道你所用的到底是哪个版本，只要看一下 BibTeX 运行时屏幕输出或草案文件 .bbl 的第一行。

B.2 创建参考文献数据库

创建参考文献数据库看起来好像要比向 thebibliography 环境中输入一串引用文献的工作量大很多。但是这样做的最大优势就是，一旦把某条目放到参考文献数据库中，也就一劳永逸了，以后所有的文章都可以引用它。即使以后需要另一种不同的参考文献样式，所有已在数据库中的条目也是可以用的，这就比用另外的格式重写 thebibliography 环境要方便得多。实际上，据我们的经验，即使正在处理的只有一篇文章，向数据库中加入一项条目，也要比向文献列表中加一项简单，因为后者需要准确而灵巧的定位，而且要考虑到标点符号和作者名字的位置。如果有了在 B.2.6 节中演示的一般性模板，数据条目的处理就会变得非常简单，而且速度很快。

在参考文献中的条目具有如下的形式

@BOOK{knuth:86a,

```

AUTHOR = "Donald E. Knuth",
TITLE = {The \TeX{}book},
EDITION = "third",
PUBLISHER = "Addison--Wesley",
ADDRESS = {Reading, Massachusetts},
YEAR = 1986 }

```

第一个单词，其前缀@确定了条目类型，我们将在下节详细解释。条目类型后接放在大括号{ }中的条目信息。其中第一项就是关键词，这也就是在\cite命令中引用的名称。在上面的例子中，其为knuth:86a。关键词可以是字母、数字以及符号（逗号除外）的任意组合。然后就是真正的索引信息，它们分放在各个域中，相互之间用逗号分开。每个域由域名，一个等号=（其两边可以有空格）以及域文本组成。上面给出的域名有AUTHOR, TITLE, PUBLISHER, ADDRESS和YEAR。域文本必须放在大括号或者双引号内。然而，如果这部分文本只由数字组成，比如上面的YEAR后的内容，那么可以不要括号或引号。

在输入条目时，有些域是不能缺少的，还有一些域则是可以省略的，而其他的域则要被BibTeX忽略。在下面与条目类型一起列出了各种域。如果缺少了某个不可少域，BibTeX运行时就会给出一条错误消息。如果使用了可省域，那么它们在参考文献中就会包含所提供的信息，但是并不一定要给出这些域的内容。而要被忽略的域可以在数据库中包含不输出的额外信息（例如注释或者论文摘要），这也是非常有用的。要被忽略的域也可以提供给其他的数据库程序使用。

在参考文献数据库中条目的一般语法为

```

@条目类型 { 关键词,
  域名 = { 域文本 }
  ....
  域名 = { 域文本 } }

```

在条目类型以及域名的名称中既可以用大写字母，也可以用小写字母，甚至是两者的混合。因此@BOOK, @book, @book都表示同样的类型。

在整个条目最外面的括号对既可以是如上所示的大括号{ }，也可以是小括号()。对于后者，一般的语法为

```
@条目类型(关键词, ... ..)
```

域文本却只能放在大括号{...}或者双引号"..."内，如上例所示。

B.2.1 条目类型

下面按字母顺序列出了标准的条目类型，并给出类型功能的简短描述，以及其可包含的不可少域和可省略域名称。我们将在下一节解释域的意义。

@article 条目为期刊或杂志上的一篇文章。

不可少域 author, title, journal, year.

可省略域 volume, number, pages, month, note.

@book 条目为有确定出版社的书籍。

不可少域 author 或 editor, title, publisher, year.

可省略域 volume 或 number, series, address, edition, month, note.

@booklet 条目为印制的有封皮的作品, 但没有出版社或赞助机构的名称。

不可少域 title.

可省略域 author, howpublished, address, month, year, note.

@conference 与下面的 @inproceedings 相同。

@inbook 条目为一本书的一部分(章, 节或某些页)。

不可少域 author 或 editor, title, chapter 和 / 或 pages, publisher, year.

可省略域 volume 或 number, series, type, address, edition, month, note.

@incollection 条目为一本书中有自己题目的一部分。

不可少域 author, title, booktitle, publisher, year.

可省略域 editor, volume 或 number, series, type, chapter, pages, address, edition, month, note.

@inproceedings 条目为会议论文集中的一篇文章。

不可少域 author, title, booktitle, year.

可省略域 editor, volume 或 number, series, pages, address, month, publisher, organization, note.

@manual 条目为科技文档。

不可少域 title.

可省略域 author, organization, address, edition, month, year, note.

@mastersthesis 条目为硕士论文。

不可少域 author, title, school, year.

可省略域 type, address, month, note.

@misc 条目为不属于其他任何类型的作品。

不可少域 没有。

可省略域 author, title, howpublished, month, year, note.

@phdthesis 条目为博士论文。

不可少域 author, title, school, year.

可省略域 type, address, month, note.

@proceedings 条目为会议论文集。

不可少域 title, year.

可省略域 editor, volume 或 number, series, address, month, organization, publisher, note.

@techreport 条目为学校或其他研究机构印制的报告。

不可少域 author, title, institution, year.

可省略域 type, number, address, month, note.

@unpublished 条目为有作者和标题的还未出版的作品。

不可少域 author, title, note.

可省略域 month, year.

在每项条目中还可以有可省略域 key 和 crossref。前者提供当没有作者信息时字母排序用的附加信息。作者信息通常就放在 author 域中,但也有可能放在 editor 域或者 organization 域中。这个 key 域与 \cite 命令中所用的识别条目的关键词没有任何关系。而 crossref 域给出另一个条目的关键词,使得两者共享某些域的信息,见 B.2.3 节。

B.2.2 域

在一项参考文献条目中可能使用的域及其含义见下面的列表。而域的形式总是为

域名 = { 域文本 } 或者

域名 = " 域文本 "

address

出版社或者研究所的地址。对于大的出版社,只要给出所在的城市就可以了。而对于小的出版社,则建议给出详细的地址。

annote 在非标准参考文献样式中可以使用评注,这样可以得到有注释的参考文献。在标准 BibTeX 样式中这个域是被忽略的。

author 给出作者的姓名,见 B.2.4 节。

booktitle

当只引用了一本书的某一部分时,应给出书的题目。参考 B.2.4 一节中有关大写方面的特殊考虑。

chapter

章节的编号。

crossref

在数据库中另一项的关键词,这样两者可以共享一些相同的域文本。见 B.2.3 节。

edition

书籍的版本,通常采用完整的首字母大写的单词表示,如 'Second'。必要时标准样式可以把它改成小写。

editor 在 B.2.4 节描述的形式中的编辑姓名。如果还有 author 域,那么用这个域给出引用书籍或作品集的编辑。

howpublished

说明出版方法的非同寻常之处。应该首字母大写,如: 'Privately published'。

institution

科技报告的举办机构。

journal

期刊或杂志的名称。对于那些相当常见的刊物名称,有相应的缩写(见 B.2.5 节)。

- key** 当没有给出作者信息时, 用此信息进行字母排序。这与 `\cite` 命令中区分条目的 **关键词** 没有任何关联。
- month** 作品出版时的月份, 或者指还未出版的作品写完时的月份。这里可以用月份(英文)名称的前三个字母进行缩写。
- note** 应加上的其他信息。首字母应该大写。
- number** 期刊、杂志、技术报告或者一系列作品中的某部的编号。期刊通常用卷和期来标识; 技术报告则由研究机构赋予一个编号; 系列书籍中的某部有时也有一个编号。
- organization**
会议或报告的主办机构。
- pages** 页码或者起始页码, 形式为 32, 41, 58, 87--101 或者 68+。最后那一种形式表示 68 页及以后所有页。单个连字符也表示页码范围, 会被转化为标准样式中的双连字符形式, 并生成一个小破折号, 如 '87-101'。
- publisher**
出版社的名称。
- school** 写作论文时所在的学术机构的名称。
- series** 一系列书或一套书的名称。当从这个系列中引用一本书时, **title** 域给出书的题目, 而可省的 **series** 给出整系列书的标题。
- title** 作品的题目, 要遵从 B.2.4 节给出的大写规则。
- type** 技术报告的类型, 例如 'Research Note'。
- volume** 期刊或者多卷书籍的卷号。
- year** 作品印刷的年代, 或者未出版作品的完成年代。年代通常要求有四位数, 例如 2000。

另外还有一些域名, BibTeX 通常就把它忽略掉。例如, 要想向数据库中加入文章的摘要, 可以用

```
abstract = {摘要文本}
```

这对于我们的应用不只是管理这个数据库时, 就非常有用了。

B.2.3 域的交叉引用

**** 本节内容只适用于 BibTeX 0.99 版本及以后版本。 ****

如果参考文献数据库中有许多条目有相同的域信息集合(例如出现在同一会议论文集集中的几篇作品), 那么就可以用 **crossref** 域来引用另一个条目中的与自己相同的域信息。例如,

```
@INPROCEEDINGS{xyz-1,
  crossref = {xyz-proceedings},
  author = {J. S. Jones},
  title = {The First Results from the {Appleville Experiment}},
  pages = {34--38} }
```

```

. . . . .
@PROCEEDINGS{xyz-proceedings,
  editor = {C. H. Kelvin},
  title = {Proceedings of the First Conference on the
           {Appleville Experiment}},
  booktitle = {Proceedings of the First Conference on the
              {Appleville Experiment}}
  year = 1991 }

```

第一项关键词是 xyz-1，利用 crossref 域从关键词为 xyz-proceedings 的第二个条目得到自己所缺少的域信息。这里缺少的域有 editor, booktitle 和 year, 这也是对会议论文集所有论文都一样的域。注意在 @PROCEEDINGS 中 booktitle 域是被忽略的，而这儿必须包含它，因为 @INPROCEEDINGS 需要这一个域的信息。

如果一个条目被其他两个或更多条目引用了，即使没有 \cite 或 \nocite 命令用其关键词做参数值，这项也会包含在参考文献中。

为了使整个系统工作正常，在数据库中那些被引用的条目必须放在引用它们的条目后面。因此我们建议把所有这些只被其他条目引用的条目放在数据库尾部。交叉索引不能嵌套。

B.2.4 特殊的域格式

对输入到 author, editor, title 和 booktitle 中的域文本必须遵从如下几条规则。当 BibTeX 处理姓名时，它会根据样式文件中的指令，首先给出作者的姓，然后是名的首字母缩写。因此如何告诉系统哪是名，哪是姓就非常重要了。对标题也有类似的问题，要根据样式和 / 或条目类型进行首字母大写，因此 BibTeX 必须知道哪些单词要被大写。

姓名

在 author 和 editor 域中的姓名既可以输入为 {名姓} 的格式，也可以是 {姓, 名} 的格式。也就是说，如果没有逗号，BibTeX 就认为最后一个首字母大写的单词是姓；否则就把逗号前面的文本当做姓。因此姓名文本 "John George Harrison" 和 "Harrison, John George" 都指的是 J. G. Harrison 先生。然而，如果一个人有复姓，中间又没有连字符隔开，那就必须采用第二种方式，或者复姓放在大括号内，如

"San Martino, Maria" 或 "Maria {San Martino}"

都表示 M. San Martino 夫人。

对于姓中首字母没有大写的辅词，例如 von 或 de，则可以采用任一种方式输入：

"Richard von Mannheim" 或 "von Mannheim, Richard"

"Walter de la Maire" 或 "de la Maire, Walter"

放在大括号内的任意文本都当做一项处理，这可以避免有时候会出现的歧义问题，例如姓名中包含逗号或单词 and。

"{Harvey and Sons, Ltd}"

如果在姓名前要加上 Junior，那会使事情变得更复杂。如果在 Jr 和名字之间有逗号，则就应该写成如下形式：

```
"Ford, Jr, Henry"
```

然而如果没有逗号，那就必须把 Jr 看成是复姓的一部分：

```
"{Filmore Jr}, Charles" 或 "Charles {Filmore Jr}"
```

****BibTeX 0.99 版本或以后版本**** 姓名中的用反斜杠命令构成的重音应该放在大括号内，并且反斜杠应该是紧接左大括号的第一个字符。通过这种方法，字母排序和在 alpha 参考文献样式中对标签的格式化都会工作正常。例如，

```
author = "Kurt G{\\"o}del",
```

```
year = 1931
```

就会生成所希望的标签 [Göd31]。重音命令被大括号包围的深度不能超过这里所示例的。

（在所有版本的 BibTeX 中，都可以把重音包含在姓名文本中，其在参考文献中的显示是相当正确的。而这里新增加的功能就是对标签和字母排序的处理。）

****BibTeX 0.99 版本及以后**** 名字中间有连字符，也能正确地缩写。因此 "Jean-Paul Sartre" 成为 'J.-P. Sartre'。（在以前的版本中，其结果为 'J. Sartre'）。

如果 author 或者 editor 域中包含不只一个姓名，那么两个姓名之间要用 and 分开。例如，

```
author = "Helmut Kopka and Daly, Patrick William" 或者
```

```
AUTHOR = {Peter C. Barnes and Tolman, Paul and Mary Smith}
```

如果 and 真的是姓名的一部分，那么就必须把整个姓名放在大括号内，见前面的示例。如果作者清单太长，无法列出所有的姓名，那么可以用 and others 表示结束。根据样式文件的规定，这将会被转化为 *et al.*。

题目

是否对题目首字母进行大写，与所用的参考文献样式有关：通常书籍题目的首字母要大写，而文章的题目则不这样做。在 title 和 booktitle 域中的文本应写成大写的形式，这样必要时 BibTeX 可以把它转化为小写形式。

在使用英语的国家，大写题目的一般规则是：题目的第一个单词，冒号后的第一个单词，以及其他单词中除冠词、没有重音的介词及连词外都要首字母大写。例如，

```
title="The Right Way to Learn: A Short-Cut to a Successful Life"
```

那些总是首字母要大写的单词（例如专用名词）应该放在大括号内。实际上只要把总是要大写的字母放在大括号内就可以了。下面两个例子是等价的：

```
title = "The {Giotto}Mission to Comet {Halley}" 或
```

```
TITLE = {The {G}iotto Mission to Comet {H}alley}
```

B.2.5 缩写

在输入域文本时通常可以使用缩写表示实际的文本。有些缩写，例如月份名称和某些标准期刊名称就可以使用缩写，当然用户也可以定义自己使用的缩写。

名称的缩写可以是字母、数字或符号的任意组合，当然下面的符号除外：

" # % ' () , = { }

缩写是用下面的命令定义的：

```
@string{缩写名称 = {文本}} 或
```

```
@string(缩写名称 = {文本})
```

其中 缩写名称 就是缩写后的形式，而 文本 是替换文本。例如，如果定义了下面这样的缩写：

```
@string{JGR = {Journal of Geophysical Research}}
```

那么下面这两条声明是一样的：

```
journal = JGR
```

```
journal = {Journal of Geophysical Research}
```

缩写名称不要放在大括号或双引号内部，否则就会把它按字面解释成域文本。在 @string 命令和缩写名称中，都不区分大小写。因此上面的缩写也可以如下定义：

```
@STRING{jgr = {Journal of Geophysical Research}} 或者
```

```
@StrinG{jGr = {Journal of Geophysical Research}}
```

在域中就可以用 JGR, JGr, JgR, Jgr, jGR, jGr, jgR 或者 jgr 来引用它。

****BibTeX 0.99 版本及以后 **** 可以在缩写之间加上 # 来把它们结合在一起。因此如果定义了

```
@string{yrbk = {Institute Yearbook}}
```

那么这个缩写就可以同其他缩写或文本结合在一起，例如

```
title = "Max-Planck~" # yrbk # 1993
```

的结果为 'Max-Planck Institute Yearbook 1993'。

月份名称是有标准缩写的，由原来名称的前三个字母组成，例如 jan, feb 等等。对于某些标准期刊，存在着一些预先定义的名称缩写。要想知道到底有哪些缩写，就需要查询相关的使用手册，因为这与安装版本有关。

@string 命令可以放在数据库中任两项条目之间，但是必须是定义在使用之前。因此把所有的缩写定义放在数据库开头就是相当合理的。

B.2.6 使用模板

向参考文献条目中输入文本似乎是一件相当复杂的工作，因为要记住那么多的事情，比如哪些域是不可少的，哪些域是可省的，它们的格式怎样，不一而足。简化这件工作的一种方法就是为经常使用的条目创建模板。所谓模板，也就是一个基本完整的条目，只是所有的域都是空的。把模板存到单独一个文件中，只要想生成一项新条目，就把它插入到文件中，然后再输入域文本。

相应于 @article 条目类型的一个比较适当的模板可以是

```
@ARTICLE{<key> ,
```

```
  AUTHOR   = {},
```

```
  TITLE    = {},
```

```

YEAR    = {},
JOURNAL = {},
  VOLUME = {},
  NUMBER = {},
  MONTH  = {},
  PAGES  = {}
}

```

其中 `<key>` 是提醒用户这里必须要输入一个关键词, 然后首先是不可少的域, 并向里缩进, 再后接可省域, 并进一步缩进。这里没有包含域 `note`, 因为虽然所有条目类型中都可以包含它, 但很少会用到。

最后提一下, `BIBTEX` 是由 Stanford 大学的 Oren Patashnik 在 Leslie Lamport 的密切协助下开发的。安装了 `BIBTEX` 后应该会生成两个文件 `btldoc.tex` 和 `btldhak.tex`, 它们包含了一些关于 `BIBTEX` 的信息。特别地, `btldhak.tex` 中有关于编写参考文献样式文件的指令。用 `LATEX` 处理这两个文件, 就可以看到其中的内容了。

B.3 扩展 `BIBTEX`

B.3.1 作者 - 年代参考文献样式

绝大多数的自然科学期刊所用的参考文献样式与 `LATEX` 所提供的标准样式不同。在这种样式中, 对其他出版物的索引是通过引用作者姓名和出版年代进行的。而且, 引用既可以是放在方括号内的, 如 `[Jones et al., 1990]`, 也可以是平铺直述的, 如 `Jones et al. [1990]`。这种样式也可以有许多变体: 例如, 可以用圆括号取代方括号, 而且名称可以用正写字体。

在标准的 `LATEX` 中, 引用是放在方括号内的一个活动编号, 如 `[10]`, 也允许逗号后面加上页码, 如 `[10, page 188]`, 但不能用文本引用。这里的编号也就是标签, 并放到参考文献中, 后接索引的文献信息。而作者-年代参考文献中没有标签, 从而直接利用作者和年代, 容易知道引用与索引之间的关系。

实际上有许多非标准的作者-年代参考文献样式可以使用, 所有这些样式都要利用一些宏包文件来重新设计 `thebibliography` 环境, 从而使它能正确解释修正后的 `\bibitem` 命令。

这其中最简单的一种样式就是由 `BIBTEX` 开发者 Oren Patashnik 所设计的, 文件名为 `apalike.bst` 和 `apalike.sty`。在这种样式中, `\bibitem` 有一个包含作者和年代的可省参数值, 如

```
\bibitem[Jones et al., 1990]{jone90}
```

`\cite` 命令的使用与通常的一样, 也就是只可以使用插入式的引用。至少有 8 个 `.bst` 文件遵从这种框架。

还有另外一组 `LATEX` 和 `BIBTEX` 样式, 称为 `newapa.sty` 和 `newapa.bst`, 由 Stephen N. Spence 开发, Young U. Ryu 对其进行了修改, 最后一次更新是在 1991 年 6 月进行

的。这是一个具有相当弹性的系统，引用既可以是插入式的，也可以是平铺直述的，而且作者的名单可以用全名，也可以缩写。这时 `\bibitem` 命令的形式为

```
\bibitem[\protect\citeauthoryear{Jones, Smith, and Harris}
        {Jones et al.}{1990}]{jone90}
```

`\bibitem` 命令的可省参数中的特殊命令 `\citeauthoryear` 用来以三个独立项的形式向 L^AT_EX 文档传递作者全名清单，缩写清单，年代。这条命令是在相应的 .sty 宏包文件中定义的。它同时提供了大量的 `\cite` 命令变体，用来显示名单与年代的组合，这既可以是插入式的，也可以是平铺直述的。有许多其他的参考文献样式，其中引人注目的 *chicago* 族，也利用了这条命令。

而对于 *newapa*，也很容易改变引用中的标点符号，比如说把 (Jones & Smith: 1990, pages 60–65) 改成 [Jones and Smith, 1990: pages 60–65]。这是利用 `\citepunct` 命令来做到的，利用它可以得到所希望的任意标点符号用法。

在参考文献样式中还有一个叫 *astronomy* 的家族，其中至少有 7 位成员。它们都应用了一条特殊命令 `\astroncite`，它实际上与前面提到的命令 `\citeauthoryear` 是一致的，但它只有两个参数值，分别相应于简短作者名单和年代。要用它们需要宏包 *astronomy.sty*。

另外一组属于作者-年代参考文献样式的应当是 *harvard* 族了。它们用 `\harvarditem` 命令取代了 `\bibitem`，但功能与 `\citeauthoryear` 一样。应用它们时，需要由 Peter Williams 和 Thorsten Schnier 所写的宏包 *harvard.sty*。这个宏包已经被更新了，从而在 L^AT_EX 2_ε 中也可以使用，现在它包含了许多新功能来定制引用。它已成为一个功能强大的宏包，选用时，相比上面所提到的其他宏包，应当优先考虑这一个。

另外，也有一些样式是在 `\bibitem` 的可省参数中应用命令 `\citename`。然而，其语法与其他的不同，而且据我们的看法，它也没有涵盖所有可能的引用形式。

Patrick W. Daly 则完成了一个更一般的宏包，它接受上面所有 `\bibitem` 中作者-年代格式，也接受数字引用框架。这个宏包名称为 *natbib.sty*，它实际上是把年代在圆括号内的可省参数取作作者-年代的基本形式，

```
\bibitem{Jones et al.(1990)}{jone90}
```

并用这种格式定义了所有其他作者-年代命令。而且也可以把所有括号和标点记号的类型，以及其他所需要的特殊功能，与 .bst 文件的名称联系起来，这样，当调用 `\bibliography` 命令时就会调用它。因此所有的使用同样参考文献样式的文档都会拥有正确的引用标点符号，甚至不用对作者部分做任何其他修改。而且对数字和作者-年代样式的选择是完全自动的。这个宏包已经相应于 L^AT_EX 2_ε 进行了更新，拥有与 *harvard.sty* 同样的功能，因此可推荐为所有参考文献样式的普适性宏包。

在 C.3.4 节给出了 *natbib.sty* 的一个简化版本，并同时给出一个基本的作者-年代参考文献样式。

所有在这里提到的样式，包括 *natbib.sty* 在内，都可以从 CTAN 文件服务器的 BibTeX 和 L^AT_EX 相应目录中取到，我们在 8.8.6 列出了这些服务器。

B.3.2 定制参考文献样式

差不多每份期刊和出版社都有自己的参考文献格式规则。这其中的差异实际上是微乎其微的,比如说此处应该用逗号还是冒号,或者编号字体为黑体还是斜体。但是,虽然差异很小,每个出版社对格式的要求却是非常严格的。

标准的 L^AT_EX 只提供了四种参考文献样式,它们的差别也就只限于排序和标签等琐碎的细节。我们可以通过修改已有的样式文件,来设计自己的 .bst 文件;然而,这需要对独特的 BibT_EX 程序设计语法有相当的了解,因为这种语法甚至会使有经验的 L^AT_EX 用户也感到迷糊。在 T_EX 文件服务器上的 BibT_EX 目录中大约有 50 个 .bst 文件,而在 L^AT_EX 相关的目录中可能有更多的此类文件。其中每个文件都是专对于某一期刊而设计的,因此无法保证能被另外的期刊接受。如果出版社要求的格式在那些地方却找不到,该怎么办呢?而且又应怎样在这些格式中搜索自己想要的格式呢?

从 custom-bib 宏包中可以得到一些帮助,这个宏包是由 Patrick W. Daly 写成,可以从文件服务器上的 L^AT_EX 相关目录中找到。这个宏包的核心就是一个通用的(或者说主要的)参考文献样式 .mbs,其中包含了相当多的针对于不同参考文献功能或选项的替换代码。要用 DocStrip 程序处理它,这个程序能根据给定的选项来包含进来替换代码。

由于样式文件中提供了大量的选项(约有 100 个),因此同时专门设计了一个菜单驱动的界面,称为 makebst.tex。当用 T_EX 或 L^AT_EX 处理这个“程序”时,它就会首先问你要读入哪个 .mbs 文件,接着就会交互构造出一个 DocStrip 批处理作业,以根据包含在 .mbs 文件中的菜单信息生成所选择的参考文献特征。这也就是说,可以有很多 .mbs 文件供选择,每一个都通过 makebst 向用户解释自己的选项集合。

事实上,merlin.mbs 是第三代公开发表的参考文献样式,它取代了原来的 genbst.mbs 以及多语言部分 babel.mbs。对其他语言的支持现在是通过另外的 .mbs 文件提供的,每个相应于一种语言,其中包含对类似于 volume, editor, edition 等等单词的翻译。

由通用参考文献样式文件所提供的一些功能如下:

- 数字或作者-年代引用;对后者,用户可以选择所使用的 \bibitem 样式;
- 索引的顺序:排序可以按引用顺序,也可以按作者姓名的字母顺序,或者根据作者-年代标签字母顺序;
- 作者姓名的格式:名加姓,名首字母加姓,姓加名首字母,只把第一作者的首字母颠倒过来,等等。
- 在 et al. 前面可以给出的作者姓名个数;
- 排版作者姓名的字样;
- 日期的位置,是否把年份放在圆括号或方括号内;
- 期刊的卷,期以及页码的格式;
- 以句子或者标题样式对文章标题进行大写;
- 在单词间用 and 还是用 &;
- 用逗号取代单词 and;
- 是否缩写 editor, volume, chapter 等;

-
- 给出起始, 结束页, 还是只给出开始页;
 - 常见期刊名称的简写形式;
 -

附录 C L^AT_EX 程序设计

L^AT_EX 2_ε 相对于以前的 L^AT_EX 版本而言, 一个主要改进之处就是为类和宏包的作者(简言之, 即 L^AT_EX 程序开发者) 提供了大力支持。绝大多数用户并没有体味到新版本相比于 L^AT_EX 2.09 的改进之处, 只是知道多了一些新的字体命令, 文档开头地方的声明现在换成了 `\documentclass`, 以及用 `\usepackage` 命令上载原来的选项文件。虽然在其他地方也有可能发现多了点新东西, 但大致的感觉就是新版本并没有带来多少新事物。然而, 对 L^AT_EX 程序开发者而言, 尤其是对曾经编写过选项文件, 或者安装过新字体框架的人员来说, 他们就会非常欣赏 NFSS 以及新的类与宏包控制功能。随着时间的推移, 基本 L^AT_EX 安装所提供的新扩展功能逐渐变得与 L^AT_EX 2_ε 密不可分, 从而想使用新功能的用户就不可避免地要把自己的系统更新到新版本上。

在 8.5 节中描述了新字体选择框架。本附录讲解设计类与宏包文件的特殊命令, 并给出了几个实用宏包的设计示例。

C.1 类与宏包文件

C.1.1 L^AT_EX 2.09 中的样式文件

类与宏包文件是新出现在 L^AT_EX 2_ε 中的概念, 它取代了 L^AT_EX 2.09 中的主样式和样式选项文件。在原来的系统中, 要想选择主样式, 就要利用声明

```
\documentstyle[选项清单]{样式}
```

这样就上载了一个名为 样式.sty 的文件, 它定义了文档所需要的全局格式。几个主要的样式为 `article`, `report` 和 `book`, 这也是 L^AT_EX 2_ε 中最重要的几个类。在样式文件的某个地方, 要用命令 `\@options` 如下处理 选项清单 中的选项:

1. 如果存在命令 `\ds@选项`, 那就执行这条命令; 否则, 就把该选项放到第二个清单中;
2. 遍历完 选项清单, 就考虑第二个清单, 对每一项, 把文件 选项.sty 输入进文档。

这一过程使得选项可以定义在样式文件内部, 也可以存贮在与选项同名的单独样式选项文件(扩展名为 .sty) 中。这样做的想法就是有些选项的代码与主样式文件无关, 从而可以存贮在外部的样式文件中。

这可能就是采取选项文件概念的初始动机, 其直接结果就是有大批的爱好者开发了数目可观的其他‘选项’, 并把它们存贮在一个文件里, 读入到文档中。通过网络服务器, 添加的选项就传播开来, 得到广泛的应用。由于其中有些只是拙劣地修补了 L^AT_EX 的内部命令, 所以不能只是简单地用 `\input` 命令把它们包含进来, 而要把文件名后缀指定为 .sty, 从而可以按照上面的第 2 点以准选项角色包含进来。但它们实际上并不是真正的选项, 只是新增加的代码或功能。

在编写主文件样式时还会出现一个新的难题。如果需要适合于某期刊的文章样式, 或者某出版社的书籍样式, 那么可以利用已有的 `article.sty` 或 `book.sty` 样式为基础,

进行必要的改动，得到新的主样式，但是不能保证在对原主样式做更新后，改动仍然有效。但是有时候一些重大的更新与对 \LaTeX 进行的改动是一致的。所以我们可以只是写一个“选项”来包含所做的改动，也可以写一个新的主样式，输入原来的样式。然而，它并不是相应于其选项清单的真正主样式。

C.1.2 \LaTeX 2 ϵ 的新概念

当 Leslie Lamport 发布 \LaTeX 时，他无论如何也不会预见到随后出现了那么多的 \LaTeX 程序。这一切现在已成为不争的事实，而且这也是该系统的魅力所在。 \LaTeX 2 ϵ 不但包容这些“外来”成员，而且实际上它进一步支持和鼓励这种趋势，其中一个明证就是在 *The \LaTeX Companion* 与 *The \LaTeX Graphics Companion*(Goossens et al.) 两书中所介绍的大量宏包。

这也就是事物发展的趋势。扩展的功能由那些需要该功能的人们设计，因为他们意识到 \LaTeX 缺少了某些对他们而言是很重要的功能。另一方面，要是把所有这些扩展的功能都加入到基本的 \LaTeX 安装中，那就会使得 90% 以上的用户虽然上载了它们，但从来不会用它们。现在解决这个问题的方法就是 \LaTeX 提供了一个基本的核心(或称内核)，然后用标准的类文件扩展其功能，接着利用那千变万化的宏包和类增加功能。

而 \LaTeX Team 的任务就是建立程序设计的方针，从而确保宏包不会与内核或者其他宏包发生不必要的冲突，而且提供一种基本的稳定性，使得那些实用的宏包将来在更新后的内核和标准类下也能正常工作。在 \LaTeX 2.09 中就缺少了这种安全机制，在那个版本中，程序设计者们被迫自己寻找门路，这实际上并不是真的程序开发。 \LaTeX 2 ϵ 在类和宏包控制方面的新功能，随同一组程序开发工具，应该在宏包内部相互作用以及对内核更新的适应等方面具有比原来更强的可靠性和持久性。

C.1.3 命令的层次

命令有许多层次，它们的安全程度也相应不同。

用户命令 (最高级命令) 在本书及其他手册中进行了描述，其名称由小写字母组成，例如 `\texttt{}`，它是被永久支持的 \LaTeX 外部定义；

类与宏包命令 其名称要稍长一些，而且大小写混杂(如 `\NeedsTeXFormat`)，主要是为程序设计人员提供的，而且也是有保障的；绝大多数是只能用在导言中的命令，但在类和宏包文件中并没有这个使用限制；

\LaTeX 内部命令 名称中包含 `@` 字符，只能用在类与宏包文件中；虽然其中有些命令对得到某些特殊效果具有重要作用，但也无法保证永远可以使用；开发人员要使用该命令，那就有可能在将来某一天自己设计的宏包失去作用；

\TeX 低级命令 名称也是由小写字母组成，而且没有 `@`；即使 \LaTeX 继续演化，这些命令的功能也应该是稳定的，但这也不是绝对的；只要有可能，就尽量避免使用它们，见后面的解释；

内部专用命令 是用于在其他人员开发的类与宏包文件内部的命令；建议所有命令都前缀大写字母(以表示宏包的名称)后接 `@`，这样可以避免与其他宏包发生冲突；例如，

在 `showkeys` 宏包中就可以有这样一条命令: `\SK@cite`.

L^AT_EX 内部命令在类和宏包文件中的应用范围到底有多大是一个令 L^AT_EX 开发人员感到迷惑的问题。因为在正式的说明书中从没有给出这些内部命令的解释, 所以总是存在着一种可能, 在将来某一天发布的新版本中不再保留这些内部命令。而我們不应杜绝使用下节所讨论的 T_EX 命令, 但我们也必须明白, 应用这些命令的同时也伴随着一定程度的风险。

L^AT_EX Team 建议的方针是, 只要有可能, 就尽量使用 L^AT_EX 高级命令。

- 要用 `\newcommand` 和 `\renewcommand`, 而不用 `\def`; 如果要使用某一个 T_EX 的定义命令 (因为调用某模板, 或者因为必须用 `\gdef` 或 `\xdef`), 那么先调用一个空的 `\newcommand` 命令, 以检测名称是否有冲突。如果无法确定命令名称是否存在, 而且该命令不是很重要的, 那么调用一条空的 `\providecommand`, 然后再调用 `\renewcommand`。现在高级命令中可以定义有一个缺省值的命令, 这使得原来经常要用低级命令的理由中缺少了重要的一条。
- 利用 `\newenvironment` 和 `\renewenvironment` 命令, 而不用 `\自己的环境` 和 `\end自己的环境` 命令对。
- 要用 `\setlength` 命令给长度和弹性长度赋值, 而不要用等号直接赋值。
- 避免使用 T_EX 盒子命令 `\setbox`, `\hbox` 以及 `\vbox`; 而要用诸如 `\sbox`, `\mbox`, `\parbox` 一类的命令。利用 L^AT_EX 2_ε 提供的可省参数值, 现在对等价的 T_EX 命令的需求大大降低, 而且 L^AT_EX 版本的命令相比 T_EX 的命令要透明得多。另外, 当用了 `color` 宏包时, L^AT_EX 的盒子仍然工作正常, 而其他命令的结果就无法预料了。
- 如果想给出错误和警告消息, 就要利用 `\PackageError` 和 `\PackageWarning`, 而不要用 `\@latexerr` 或 `\@warning`; 前面两条命令也同时告诉用户消息的来源, 而不是只把它们标为 L^AT_EX 消息。
- 我们不会建议你只使用 `ifthen` 宏包 (7.3.5 节) 中的 `\ifthenelse` 命令, 以代替 T_EX 的条件命令。但是似乎用这个宏包可以简化对条件的应用, 而且符合 L^AT_EX 的语法。

本书所有的例子都用的是这个宏包, 这就不需要对相应的 T_EX 命令再进行解释了。遵从上述规则以及与之类似的规则, 有助于在将来即使 L^AT_EX 内核进行了更新, 宏包也能继续保持有效的功能。

C.1.4 T_EX 命令

为什么要避免用那些基本的 T_EX 命令呢? 在定义一条新命令时, 用 `\def` 至少不会比用 `\newcommand` 差, 而且有时候还必须用前者。那么在将来出现的 L^AT_EX 3 中这条命令有可能被去掉吗? 基本命令 (原语) 是所有 L^AT_EX 风味得以建立的构造模块, 因此它们一定会维持不变的。

而这并不是这样做的关键所在。基本 T_EX 命令构成了所有格式的基石, 从而所有用它们直接定义的命令的功能就永远与开发人员所期望的一样。然而, 等价的 L^AT_EX 工具所能做的事情却会随着时间的发展而增多。例如, `\newcommand` 命令可以检测新定义的命令是否与已有命令发生冲突。而且, 以后完全有可能会加进一种调试机制, 它可以跟

踪所有的重定义；而用 `\def` 定义的命令则是排它的。而且现在的 L^AT_EX 2_ε 中也有一种机制，它可以跟踪所有中级和高级命令的文件输入。

L^AT_EX 2_ε 中的牢固命令是另外一个容易让低层次程序开发人员误入歧途的例子。有很多命令本质上是脆弱的，也就是说，当把它们用做其他命令的参数值时，会过早被解释，可如果给它们前缀 `\protect`，一般可以使其变得牢固。在 L^AT_EX 2.09 中，有几条脆弱命令在定义时采用的是一种牢固方式，即定义中包含了 `\protect`，例如 L^AT_EX 的标志命令：

```
\def\LaTeX{\protect\p@latex}
\def\p@latex{...}
```

真正的定义是在内部的 `\p@latex` 中，并不是在外部的 `\LaTeX` 中。由于如此的标志定义中有很多缺陷，因此有几个宏包引入了一个改进的版本。它们只是重定义 `\LaTeX`，从而使这条命令变成脆弱的了；因此聪明的方法是重定义 `\p@latex`，这就利用了隐藏在 L^AT_EX 2_ε 后台的结果，但是命令却以一种完全不同（而且更好）的方式变得牢固了。（顺便说一下，现在 L^AT_EX 标志的内部定义已有了很大的改进。）

虽然我们希望只使用正式发布的 L^AT_EX 命令，但在很多情形中，我们必须用 L^AT_EX 内部命令或者 T_EX 基本命令。在目前阶段，为了得到一个工作稳定的宏包，我们就必须考虑在将来可能会不兼容的风险。而且如果有等价的高级命令可用，我们就不应冒这个风险。

C.2 L^AT_EX 2_ε 程序设计语言

本节描述的所有命令都是在 L^AT_EX 2_ε 中新增加的。虽然对类和宏包文件而言，这些命令并不是非常重要的，但它们确实扩展了类与宏包的用途，并为类与宏包在使用时的正确性提供了更强的保证。

C.2.1 文件识别

有三条命令可用来测试类或宏包插入时所处的外部环境是否正确。其中第一条为

```
\NeedsTeXFormat{ 格式 }[ 版本 ]
```

在类或宏包中的第一条语句就应该是所需要的 T_EX 格式声明。虽然到现在为止，已有很多不同名称的 T_EX 格式，但只有名为 `LaTeX2e` 的格式才认识这条声明。而在所有其他格式中调用这条命令，都会给出错误消息：

```
! Undefined control sequence.
```

```
1.1 \NeedsTeXFormat
```

```
{LaTeX2e}
```

在这种情况下，上述消息实际上就给出了一种提示信息。

在 L^AT_EX 2_ε 中，对这条命令而言，可能更有用的地方是它的可省参数值 `版本`，该参数的形式必须为表示发行日期的 `yyyy/mm/dd`。如果一个宏包利用的功能是在某一个版本中才引进的，那就必须给出这个日期，因此如果用的是一个更早的 L^AT_EX 2_ε 版本，就会显示出

一条警告。例如,在 L^AT_EX 2_ε 起初的测试版本中并没有提供命令 `\DeclareRobustCommand`, 只是在 1994 年 6 月 1 日正式发行时才有了这条命令。因此使用了这条命令的宏包就应该以下面这条语句开头:

```
\NeedsTeXFormat{LaTeX2e}[1994/06/01]
```

这里日期的形式是很重要的,必须有零和斜杠。

这条声明并不是只限于用在类和宏包文件中,也可以在文档开头调用它,以保证用正确的 L^AT_EX 版本处理该文档。然而调用的位置必须是在导言中。

下面两条命令用来标明类或宏包文件自身:

```
\ProvidesClass{类}[版本]
```

```
\ProvidesPackage{宏包}[版本]
```

在这两条命令中,版本都是由三部分组成的:日期,版本号以及附加信息。日期与上面的格式相同,而版本号可以是任何没有空格的标志,附加信息可以有或没有空格的文本。例如,

```
\ProvidesPackage{shortpag}[1995/03/24 v1.4 (F. Barnes)]
```

L^AT_EX 只会检查其中的日期部分,也就是说,如果使用了 `\usepackage` 命令并给出了日期,那么 L^AT_EX 就会对两处的日期进行比较。如果调用了 `\listfiles` 命令,就会显示出版本号和附加信息部分。

`\documentclass` 和 `\usepackage`(以及 `\LoadClass` 和 `\RequirePackage`) 命令都可以包含一个可省参数,以指定类或宏包可接受的最早发行日期。例如,当声明为

```
\documentclass[12pt]{article}[1995/01/01]
```

这时如果使用的 `article` 类文件中包含

```
\ProvidesClass{article}[1994/07/13 v1.2u
```

```
Standard LaTeX document class]
```

那么就会显示出一条警告消息。对于 `\usepackage` 和 `\ProvidesPackage` 命令,也是同样的机理。

版本检测机制使得文档可以索取处理自己的合适版本的类和宏包文件。当然这里要假设所有后来的版本都与以前的版本完全兼容。

对于那些用 `\input` 命令上载的普通文件,还有一条识别命令:

```
\ProvidesFile{文件名}[版本]
```

此时不会检测名称或版本,但是利用 `\listfiles` 可以列出这两部分信息。

C.2.2 上载其他类和宏包

在主文档文件中,类的读入是利用初始化命令 `\documentclass` 来实现的,而宏包用的则是 `\usepackage` 命令。而在类和宏包文件内部,就必须使用下述命令:

```
\LoadClass[选项]{类}[版本]
```

```
\RequirePackage[选项]{宏包}[版本]
```

其中第一条命令可使得一个类文件上载另一个类文件,并且需要的话,可以给出选项;而第二条命令使得类和宏包文件上载其他的宏包。在任何类文件中只能有一条 `\LoadClass`

命令, 该命令不能用在宏包文件中。这两条命令都可以用在文档文件中。其中的宏包参数值可以是由几个宏包名称组成的清单, 中间用逗号分开。

可省的版本参数与相应的 `\Provides..` 命令之间的关系在前一节中做了介绍。我们下面将介绍选项参数的处理方式。

C.2.3 选项的处理

在类和宏包中都可以有选项, 其定义方式为

```
\DeclareOption{选项}{代码}
```

其中选项就是选项的名称, 而代码就是选项要执行的指令集。在 L^AT_EX 内部, 实际上创建了一条名称为 `\ds@选项` 的命令。通常这些代码并不做任何事, 只是设置一些标志, 或输入一个选项文件。(`\RequirePackage` 不可以用在选项代码中!) 在 `article.cls` 文件中的两个示例为

```
\DeclareOption{fleqn}{\input{fleqn.clo}}
\DeclareOption{openbib}{\setboolean{@openbib}{true}}
```

可以用 `\DeclareOption*` 定义一个缺省选项, 这条命令并不需要选项名称, 只是指定适用于所有被调用的未定义选项的执行代码。

有两条特殊命令, 可以用在缺省选项定义的代码中:

```
\CurrentOption 由正在被处理的选项名称组成;
\OptionNotUsed 把 \CurrentOption 声明为未处理的。
```

例如, 若想有一个类文件, 模拟 L^AT_EX 2.09 对所有未定义选项上载同名的 `.sty` 文件时的行为, 可以如下定义:

```
\DeclareOption*{\InputIfFileExists{\CurrentOption.sty}%
{}{\OptionNotUsed}}
```

这样就会首先检测是否存在指定名称的 `.sty` 文件, 如果不存在, 就把选项声明为没有使用的。要求的选项没有使用(处理)的话, 就会列在一条警告消息中。

接下来就用下面的命令处理选项:

```
\ExecuteOptions{选项清单}
\ProcessOptions
\ProcessOptions*
```

其中 `\ExecuteOptions` 会为选项清单中的每个选项调用 `\ds@选项` 命令。在默认方式下通常就是建立起特定的选项配置。 `\ProcessOptions` 按照所有选项定义的顺序执行指定的选项, 然后删除它们。这也就是说这条命令只能执行一次。有星号的命令功能类似, 只是它是按调用时指定的顺序执行。为了与 L^AT_EX 2.09 样式兼容, 现在仍然保留了命令 `\@options`, 它只是 `\ProcessOptions*` 命令的另一个名称而已。

也可以用下面的命令为类或宏包定义选项:

```
\PassOptionsToClass{选项}{类名}
\PassOptionsToPackage{选项}{宏包名称}
```

其中选项是指定类或宏包文件可以识别的一串合法选项。这两条命令可以用在其他选项的定义中。最常见的用法就是把缺省选项传递给另一个类，如下例所示：

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
```

这里所用的类或宏包必须在稍后用 `\LoadClass` 或 `\RequirePackage` 命令来上载。

如果类和宏包文件的缺省选项并没有用 `\DeclareOption*` 进行改变，那么处理未定义的被调用选项的标准程序如下：

- 所有在 `\documentclass` 语句中的选项标记为全局的；认为它要应用于后面所有宏包，但用 `\LoadClass` 上载的类除外；如果在主类中没有定义该选项，不会给出错误或警告消息；
- 所有其他语句（包括 `\LoadClass` 和 `\PassOptionsTo..`）给出的选项都是局部的；如果在相应的类或宏包中没有定义该选项，会给出一条错误消息；
- 当所有宏包都读进来后，如果某一个全局选项还从来没有用过（从没有被定义），就会给出一条警告消息；
- 所有的选项（无论是全局选项，还是局部选项），都按照它们的定义顺序执行，除非调用了 `\ProcessOptions*`。

C.2.4 延迟处理

有时候，为了得到某种特殊效果，或者避免与其他宏包发生冲突，希望有些命令是在类或宏包结束处执行，或者在文档的开头或结尾处执行。可以用下面的命令实现这种效果：

```
\AtEndOfClass{ 命令集 }
\AtEndOfPackage{ 命令集 }
\AtBeginDocument{ 命令集 }
\AtEndDocument{ 命令集 }
```

前两条命令把命令集保留到类或宏包结尾时才执行。在开始时读入的配置文件中可以利用这些命令，但这些命令由在结尾处所应进行的修改组成，这样在结束时就不会由于缺省方式而被重写。后两条声明把命令集分别保留到 `\begin{document}` 和 `\end{document}` 中执行。上述所有命令都可以多次调用，这样命令集执行时就会按照调用的顺序进行。

保存在 `\AtBeginDocument` 中的命令集是会准确地插入在导言的处理流中，但它是在 `\begin{document}` 命令已几乎做完自己该做的事情之后。因此可以认为命令集是正文的一部分，但是那些只能用在导言中的命令也可以用在命令集中。

C.2.5 牢固的命令

在 7.3 节中我们详细讲述了如何使用 `\newcommand` 定义新的命令，使用 `\renewcommand` 重定义命令，以及使用 `\providecommand` 临时创建命令。实际上还有另外两条语法相同的定义命令的语句：

```
\DeclareRobustCommand{ 命令名 }[ 参数个数 ][ 可省参数 ]{ 定义 }
```

用来定义或重定义一个名为 `\命令名` 的命令，而且它是牢固的，也就是说它可以用做其

他命令的参数值，前面不需前缀 `\protect`。如果这里定义的命令已经存在了，就会向抄本文件写入一条消息，并覆盖原来的定义。

另外一条命令可以用来检测 `\命令名` 的当前定义。

`\CheckCommand{命令名}[参数个数][可省参数][定义]`

如果出现命令的定义与 `\命令名` 定义不同，或者参数个数不同等情况，就会给出一条警告消息。这可以用来检查已上载的宏包是否修改了某些重要的定义，从而确认系统的状态是否为我们所希望的样子。

`\DeclareRobustCommand` 和 `\CheckCommand` 命令都可以用在文档的任何地方。

C.2.6 有短参数值的命令

通常在用户定义命令的参数值中可以包含用 `\par` 或空行表示的新段落。按 T_EX 的行话来说，这些命令都是“长”的。这就不是用 `\def` 定义命令的标准行为，因为用它定义的命令必须是短的，这样可以检测是否遗漏了右大括号。

到了 1994 年 6 月 1 日，新发行的 L^AT_EX 2_ε 提供了所有定义命令的星号形式：

```
\newcommand*           \renewcommand*
\newenvironment*       \renewenvironment*
\providecommand*
\DeclareRobustCommand* \CheckCommand*
```

上述命令创建的命令都具有“短”参数，从而行为与 `\def` 相同。

我们建议总是用星号形式命令来定义新的命令，除非有足够的理由要使某些参数为“长”的，即参数中包含段落。长参数应当是例外，而不是规则。

C.2.7 给出错误和警告消息

在设计类和宏包时，也应当使它们具有自己的错误和警告消息。这对帮助我们辨别到底是哪个文件发出这条消息是非常有用的。

错误消息是用下列命令生成的：

```
\ClassError{类名}{错误消息文本}{帮助}
\PackageError{宏包名}{错误消息文本}{帮助}
```

其中 `错误消息文本` 就是显示在监视器或抄本文件中的消息，而 `帮助` 就是当用户反映为 H 时显示的进一步文本。如果文本中包含命令名，而且要按原样显示，那就必须前缀 `\protect`；空格是用 `\space` 生成的，新行用 `\MessageBreak` 开始。例如，

```
\PackageError{ghost}{%
  The \protect\textwidth\space is too large\MessageBreak
  for the paper you have selected}
  {Use a smaller width.}
```

就会生成如下错误消息：

```
! Package ghost Error: The \textwidth is too large
(ghost)                for the paper you have selected.
```

See the ghost package documentation for explanation.

Type H <return> for immediate help.

当 L^AT_EX 停下来等用户给出一个反应时，若按 D.1 节中描述那样输入 H<回车>，就会得到

Use a smaller width.

在类和宏包中也可以按类似的方法给出警告消息。差别就在于后者没有 帮助 文本，而且处理过程也不会停下来等待反应。可以包含警告消息出现时在输入文件中所处的行号。

```
\ClassWarning{类名}{警告消息文本}
\ClassWarningNoLine{类名}{警告消息文本}
\PackageWarning{宏包名}{警告消息文本}
\PackageWarningNoLine{宏包名}{警告消息文本}
```

例如，当定义了

```
\PackageWarning{ghost}
  {This text is haunted}
```

就会得到了消息

Package ghost Warning: This text is haunted on input line 20.

而且处理不会停下来。警告消息可以用 \MessageBreak 分成几行，这一点与错误消息中的类似。

此类型的最后两条命令是

```
\ClassInfo{类名}{信息文本}
\PackageInfo{宏包名}{信息文本}
```

它只把文本写到抄本文件中，不会显示在监视器上。从其他角度来看，就如同没有 NoLine 的警告消息。

C.2.8 输入文件

不是类和宏包的文件也可以输入到文档中，但这样做的时候，通常必须保证文件是存在的。或者，根据文件是否存在来决定要采取的进一步行动。这一目标是用如下命令实现的。

```
\IfFileExists{文件名}{真}{假}
\InputIfFileExists{文件名}{真}{假}
```

这两条命令都会在 L^AT_EX 文件搜索路径中看看有没有指定 文件名的文件，如果找到了文件，就执行 真 中的命令，否则就执行 假 中的命令。而且，在 \InputIfFileExists 命令中，执行了 真 后还会读入该文件。

这些命令并不限于只用在导言中，也不限于只用在类或宏包文件中。实际上，通常的 \input 命令就是用它们定义的。

许多特殊的类利用这些命令读入配置文件。例如，在类 ltxdoc 中包含语句

```
\InputIfFileExists{ltxdoc.cfg}
  {\typeout{Local config file ltxdoc.cfg used}}
  {}
```

这一语句就放在 `\ProcessOptions` 前面。这样可以有一个局部配置文件，其相应于欧洲安装版本，指定

```
\PassOptionsToClass{a4paper}{article}
```

而不必修改要用 `ltxdoc` 类处理的文件。

C.2.9 检测文件

我们在此描述两条跟踪使用文件的命令，它们并不是程序设计的一部分。其中第一条命令就是我们在 8.1.1 节已提到的命令

```
\listfiles
```

这条命令可以放在导言中，甚至 `\documentclass` 命令的前面。在处理过程结束后，它会生成并显示出所有输入文件的清单，同时包括文件的版本和发行数据。用这种方法，我们就可以得到所有被包含进来的文件记录，当要把一个文件送到另外的地方，用不同的安装版本进行处理时，这一记录信息就可能非常有用。由于非标准文件也有可能被包含进来，那么从上而的清单中就很容易识别出来。

例如，输入下面这个简单的文档文件：

```
\documentclass{article}
\usepackage{ifthen}
\listfiles
\begin{document}
  \input{mymacros}
  This is \te.
\end{document}
```

就会得到如下的清单

```
*File List*
article.cls 1994/07/13 v1.2u Standard LaTeX document class
size10.clo 1994/07/13 v1.2u Standard LaTeX file (size option)
ifthen.sty 1994/05/27 v1.0i Standard LaTeX ifthen package (DPC)
mymacros.tex
*****
```

在这种情形中，局部文件 `mymacros.tex` 中不包含版本信息，因为其中没有用 `\ProvidesFile` 命令。

如果要把上而这个文件送到其他地方进行处理，那么该如何处理像上面 `mymacros.tex` 那样的局部文件呢？当然可以与主文件一起寄送该局部文件，这就需要告诉收件人更多

指令，以确定该如何行事。另外一种方法就是为了寄送文件，把局部文件内容直接包含在主文件中。对于宏包文件，这样做可就不是那么容易了，因为内部命令中有 \circ 符号，会造成一些麻烦，从而不能正确处理一些选项。现在 L^AT_EX 2_ε 提供了如下环境：

```
\begin{filecontents}{文件名}
    文件内容
\end{filecontents}
```

或

```
\begin{filecontents*}{文件名}
    文件内容
\end{filecontents*}
```

这个环境可以用在文档开头，即 `\documentclass` 命令的前面。这个环境首先会检测系统中是否存在一个文件，名为 `文件名`，如果不存在，它就会把 `文件内容` 照原样写到那个名称相应的文件中。该文件可以是一个宏包，随后要用 `\usepackage` 上载它。利用这种方法，缺少的非标准文件就可以与主文档文件一起寄送了。

我们推广上面那个简单例子，在开头部分输入

```
\begin{filecontents}{mymacros}
\newcommand{\te}{the end}
\end{filecontents}
```

那么新生成的 `mymacros.tex` 内容为

```
%% LaTeX2e file 'mymacros'
%% generated by the 'filecontents' environment
%% from source 'mydoc' on 1994/09/27.
%%
\newcommand{\te}{the end}
```

注意 `filecontents` 环境加进了一些注释行，说明新文件来自于何处。如果不希望加进这些注释行，那可以用 `filecontents*` 环境。

C.2.10 兼容模式

为了使得专门为 L^AT_EX 2.09 编写的老文档也可以用 L^AT_EX 2_ε 进行处理，现在存在着一个 **兼容模式**，它用 `\documentstyle` 取代 `\documentclass`。这样整篇文档就可以只遵从原来的标准，但不能使用 L^AT_EX 2_ε 的所有新功能。

然而，兼容模式仍旧上载新的类文件，而不是原来的样式文件，因为这些样式文件将来可能会不存在。该模式首先查找后缀为 `.cls` 的文件，只有当该文件不存在时，它才会上载 `.sty` 文件。这就是为了迁就原来的非标准样式，使其功能就像类一样。

类似于 `article` 这样后缀为 `.cls` 的标准类文件，甚至用 `\documentstyle` 也可以上载。但是这时其功能必须与将要废除的 `.sty` 文件功能一样。实际上，它们是有很多差别的，例如如何设置页边和文本尺寸。不但如此，我们要求兼容模式的输出也必须与用 L^AT_EX 2.09 以及 `article.sty` 处理时完全一样。为了做到这一点，在兼容模式中把

`@compatibility` 布尔变量 (7.3.5 节) 设置成 (真), 这样任一类或宏包都可以检测该模式。检测的形式为

```
\ifthenelse{\boolean{@compatibility}}{真}{假}
```

其中 真 表示只适用于兼容模式的命令, 而 假 表示那些可以用在真正 L^AT_EX 2_ε 模式中的普通命令。

对于所有的标准类文件, 仍然存在着 .sty 文件, 例如 `article.sty`, 只不过内容是空的, 它只是给出一条警告消息, 并上载类文件。这是为了兼容那些旧宏包, 因为它们有可能显式地上载这些文件。

C.2.11 有用的内部命令

尽管在 C.1.3 节中的方针建议最好避免用 L^AT_EX 内部命令, 但是其中有些命令是非常基本的, 它们组成了 L^AT_EX 内核和许多标准宏包的构造模块。由于它们终究还是内部命令, 因此无法保证在以后的更新中还是存在的。然而, 如果真的没有了它们, 那么就要对由 L^AT_EX Team 所提供的大量有趣的扩展功能的宏包进行全面的大检修。我们这里只是为那些勇敢的用户们简单地介绍一下这些命令。在 L^AT_EX 2_ε 和 2.09 中都存在这些命令:

```
\@namedef{命令}{定义}
```

```
\@nameuse{命令}
```

就会定义并执行名为 \命令 的新命令, 其中命令名称中并不包含反斜杠。这个名称中可以包含任意字符, 即使通常在命令名称中禁止使用的字符也可以。

```
\@ifundefined{命令}{真}{假}
```

如果 \命令 不存在, 就执行 真, 否则执行 假。同样, 这里 命令 中也不包含反斜杠, 而且任何字符都可以出现在命令名称中。这个检测语句通常用来有条件地定义命令, 其功能现在已经被 `\providecommand` 取代。也可以用它确定正在使用的主类是否是 `article`, 即用 `\@ifundefined{chapter}{..}{..}` 可以检测 `\chapter` 命令是否存在, 从而测试出当前的类。

```
\@ifnextchar 字符 {真}{假}
```

用来检测下一个字符是否是给定的 字符, 如果是的话, 执行 真, 否则执行 假。这条命令通常用来定义有可省参数的命令, 这时 字符 就是 [。新扩展的 `\newcommand` 命令用高级方法得到了这一效果。

```
\@ifstar{真}{假}
```

用来检测下一个字符是否是星号 *, 如果是的话, 就执行 真, 否则执行 假。可以用它来定义带星号的命令和环境, 这是高级命令做不到的。

```
\@for \对象 := \列表 \do {命令}
```

其中 \列表 就是一条命令, 被定义成一串用逗号分开的元素, 而 \对象 就相继取每个元素的值, 并对每个元素执行一次 命令 代码。例如,

```
\newcommand{\set}{start,middle,end}
```

```
\@for \xx := \set \do {This is the \xx. }
```

就会显示出 'This is the start. This is the middle. This is the end.'

C.2.12 有用的 T_EX 命令

L^AT_EX 中许多复杂的功能以及宏包都只能用 Plain T_EX 命令开发出来。这些命令的描述不但可以在 Knuth 的 *The T_EXbook*(1984 年) 一书中找到, 而且还有一本相当好的参考书, 那就是 Eijkhout 的 *T_EX by Topic, a T_EXnician's Reference* (1992 年)。

我们打算要把本书写成 T_EX 手册; 不但如此, 而且常用的出现在许多宏包以及后面示例中的 T_EX 命令数目也很少。只要给出一个简短的描述, 就对理解这些命令的作用非常有帮助。真正的 T_EX 专家和 T_EX 技术人员可以略过这一节。

`\def\命令 #1#2..{定义}`

是 T_EX 中标准的定义命令。它等价于 `\newcommand`, 只是它不会检查是否有名称冲突, 而且参数的指定也不同。例如, 显示科学记数法的命令 `\Exp` 可以如下定义:

`\def\Exp#1#2{\ensuremath{#1\times 10^{#2}}}`

或者

`\newcommand{\Exp}[2]{\ensuremath{#1\times 10^{#2}}}`

对于这两种定义方式, `\Exp{1.1}{4}` 的结果都是 1.1×10^4 。然而, `\def` 还可以做得更多。它可以把参数放在一个模板中, 例如

`\def\Exp#1(#2){\ensuremath{#1\times 10^{#2}}}`

这样可以得到更方便的记号 `\Exp1.1(4)`, 而这是 `\newcommand` 命令做不到的。当定义命令时, 不知道 (或不关心) 是否已存在同名命令时, 或者要用模板时就用 `\def`。有可省参数的命令实际上就是用模板定义的。

`\gdef` `\edef` `\xdef`

是 `\def` 的变形; 第一条命令给出一个全局定义, 所得命令即使在当前环境或者 `{...}` 括号对外面也仍然有效; 第二条命令是一个展开的定义, 其中任何命令都具有自己本身的意义, 而并不是把命令插入在定义中; 最后那条是前面两条的组合, 即展开的全局性定义。

`\noexpand` `\expandafter`

控制命令在定义和执行时的展开。在 `\edef` 中定义部分的任何命令都要被展开 (插入其含义), 除非其前缀 `\noexpand`。相反的效果可以用 `\expandafter` 得到, 这条命令跳过后接命令, 展开下一条命令, 然后执行被跳过的命令。这就是相当深奥的 T_EX 技术了, 我们最好还是用上面那条 `\Exp` 命令来演示一下结果。

`\newcommand{\mynums}{1.1(4)} \expandafter\Exp\mynums`

就等价于 `\Exp1.1(4)`, 而这与 `\Exp\mynums` 并不等价; 在执行 `\Exp` 之前先把 `\mynums` 展开成 `1.1(4)`。

`\let\命令一 = \命令二` 或 `\let\命令一 \命令二`

使得 `\命令一` 取 `\命令二` 的当前含义。这通常用来在重定义命令前保存其原来定义, 从而可以同时使用其原来定义。

`\relax`

绝对不做任何事, 但通常用来插在应该有些什么东西而我们又不想有内容的地方。

```
\if条件真 \else假 \fi
```

就是 \TeX 中条件语句的形式。有许多不同形式的条件，我们这里就不一一介绍了，但常见的应用就是等价于 \LaTeX 的布尔变量命令：

```
\newif\if标志      = \newboolean{标志}
\标志true          = \setboolean{标志}{true}
\标志false         = \setboolean{标志}{false}
\if标志..\else..\fi = \ifthenelse{\boolean{标志}}{..}{..}
```

对那些经常用这些语句的人而言， \TeX 形式要紧凑一些，但它并不与一般的 \LaTeX 行事方式一致。

```
\ifcase 数 文本0 \or 文本1 \or...\fi
```

会根据 数 的值来决定执行哪个 文本。

```
\endinput
```

终止对当前文件的输入。这并不是必需的方式，但所有的文件都用它结束不失为一个好的程序设计习惯。在主文档中不必用它，因为 `\end{document}` 可以得到同样的效果。

C.3 宏包示例

我们下面给出几个示范性宏包，以说明前面一节给出的程序设计命令的使用方法。

C.3.1 修改文本尺寸

在标准 \LaTeX 类中，要根据 `\documentclass` 中指定的尺寸选项（如选项 `a4paper` 或 `legalpaper`）来设置文本尺寸参数 `\textwidth` 和 `\textheight`，而且同时调整页边距，使得文本水平和竖直居中。`\textwidth` 的值是有限制的，每行上最多有 60-70 个字符，这主要是出于排版中达到最佳视觉效果考虑。

而有时候我们想去掉这个限制，充分利用纸张的最大幅度。只有通过多次尝试，我们才有可能找到恰当的参数组合，匹配特定的纸张尺寸。如果有一个宏包可以帮你做到这一点，那就太好了。

下面给出的 `fullpage` 宏包利用了 `\paperwidth` 和 `\paperheight` 的值（这两个值是要根据纸张尺寸选项进行设置的），以生成两边只有一英寸页边的页面。它甚至可以考虑到当前页面样式，从而为必要的书眉和脚码留下空间。如果给出必要的选项，甚至可以得到只有 1.5cm 窄的页边。

这个宏包的开头部分就声明自己需要 $\text{\LaTeX} 2_{\epsilon}$ ，然后标明自己的身份。宏包信息由预定义格式的日期、版本号以及作者姓名大写首字母组成。由于需要条件判断，接着就加载了 `ifthen` 宏包。

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{fullpage}[1994/02/15 1.0 (PWD)]
\RequirePackage{ifthen}
```

下面就需要准备好选项。选项有 `in` 和 `cm` 分别相应于 1 英寸和 1.5 厘米（1 厘米就太窄了）的页边。要用一个特殊长度存贮页边长度值。这个长度是私有的内部命令，遵从

266 页上的约定。

```
\newlength{\FP@margin}
\DeclareOption{in}{\setlength{\FP@margin}{1in}}
\DeclareOption{cm}{\setlength{\FP@margin}{1.5cm}}
另外，四个标准的页面样式也做为选项名。它们将会设置两个内部布尔变量和页面样式。
\newboolean{FP@plain}
\newboolean{FP@empty}
\DeclareOption{plain}{\setboolean{FP@plain}{true}
                     \setboolean{FP@empty}{false}
                     \pagestyle{plain}}
```

```
\DeclareOption{empty}{\setboolean{FP@plain}{true}
                     \setboolean{FP@empty}{true}
                     \pagestyle{empty}}
\DeclareOption{headings}{\setboolean{FP@plain}{false}
                          \setboolean{FP@empty}{false}
                          \pagestyle{headings}}
\DeclareOption{myheadings}{\setboolean{FP@plain}{false}
                           \setboolean{FP@empty}{false}
                           \pagestyle{myheadings}}
```

最后执行选项的缺省集，再处理选定的选项，在这种情形中，是按指定的顺序进行的。之所以这样做，就是如果给出了不只一个页面样式，最后那个选项起作用。

```
\ExecuteOptions{in,plain}
\ProcessOptions*
```

下面开始进行计算。首先对于 plain 和 empty 样式，没有书眉行，因此把相应的参数取值为零。这时 FP@plain 为〈真〉。然后把为脚码行保留的空间设为零（此时 FP@empty 为〈真〉）。对于 headings 和 myheadings，这些空间维持不变，因为在这些样式中通常第一页就是 plain 页（有页脚）。

```
\ifthenelse{\boolean{FP@plain}}
  {\setlength{\headheight}{0pt}
   \setlength{\headsep}{0pt}}{}
\ifthenelse{\boolean{FP@empty}}
  {\setlength{\footskip}{0pt}}{}

```

在页边，书眉和脚码空间已设置好后，就可以进行实际的计算。注意打印机会在左边和上边留下 1 英寸的边界，因此要从 \topmargin 和 \oddsidemargin 中减去这个长度。

```
\setlength{\textwidth}{\paperwidth}
\addtolength{\textwidth}{-2\FP@margin}
\setlength{\oddsidemargin}{\FP@margin}
\addtolength{\oddsidemargin}{-1in}
```



```
\setlength{\evensidemargin}{\oddsidemargin}
```

```
\setlength{\textheight}{\paperheight}
```

```
\addtolength{\textheight}{-\headheight}
```

```
\addtolength{\textheight}{-\headsep}
```

```
\addtolength{\textheight}{-\footskip}
```

```
\addtolength{\textheight}{-2\FP@margin}
```

```
\setlength{\topmargin}{\FP@margin}
```

```
\addtolength{\topmargin}{-1in}
```

注意 `\paperheight` 和 `\paperwidth` 是被纸张尺寸选项设置成恰好等于纸张的完全尺寸的。如果这种指定是错误的，那么当然最后的结果也不会正确。

至此就完成了宏包文件 `fullpage.sty` 的所有代码。下面是调用它的一个示例：

```
\documentclass[a4paper,12pt]{article}
\usepackage[headings]{fullpage}
. . . . .
```

C.3.2 重新设计书眉和脚码

L^AT_EX 用户经常要做的一件事，可能就是调整每页上书眉和脚码的显示了。标准的 L^AT_EX 虽然提供了有限数目的页面样式(3.2 节)，但是通常是不够用的。它们中间最有弹性的就是 `myheadings` 页面样式了，它可以让作者利用 `\markright` 和 `\markboth` 来确定活动书眉的文本内容。然而，包括字体样式和页码位置等一般性格式仍然由 L^AT_EX 决定。

我们下面给出一个例子，说明如何为一个工程文档修改书眉，使得其包含特定的信息。除了标题和节的简写形式、章节号与页码外，还可以有文档序列号、日期，以及版本号。如果能设计一个广义文档页面样式，可以在页面定义外面指定这些条目，那就非常方便了。

由于这里的编码只适用于 `article` 类，因此我们就创建一个新类，名称为 `dohead`，它输入 `article.cls`，然后定义一个新的页面样式。实际上这个类中可以包含更多的其他特殊功能，我们的新页面样式只是其中一个。

我们还是首先声明所需要的 T_EX 版本，并标明类文件。

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{dohead}[1994/09/28 1.0 (PWD)]
```

这个类没有自己的新选项，只是把指定的选项传递给后台的 `article` 类。然而，它自己自动选择 `12pt` 和 `a4paper` 选项。

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
\LoadClass[12pt,a4paper]{article}
```

接下来就是用来输入页眉中四部分信息（短标题、日期、序列号，以及版本）的命令。其中每条命令都把自己的参数存贮在一条内部的 `\DH@` 命令里，这几条内部命令起初

设成空。最后，声明这些输入命令只能用在导言中，因为如果在正文已开始后再调用它们，就会导致灾难性的后果。

```
\newcommand{\DocTitle}[1] {\renewcommand{\DH@title}{#1}}
\newcommand{\DocDate}[1] {\renewcommand{\DH@date}{#1}}
\newcommand{\DocNumber}[1] {\renewcommand{\DH@number}{#1}}
\newcommand{\DocVersion}[1] {\renewcommand{\DH@version}{#1}}
\newcommand{\DH@title}{} \newcommand{\DH@date}{}
\newcommand{\DH@number}{} \newcommand{\DH@version}{}
\@onlypreamble{\DocTitle} \@onlypreamble{\DocDate}
\@onlypreamble{\DocNumber} \@onlypreamble{\DocVersion}
\@onlypreamble 是一条 LATEX 的内部命令。
```

我们下面定义新的页面样式 dochead，也就是说我们必须创建一个叫 \ps@dochead 的命令，它是由 \pagestyle{dochead} 执行的。这条命令要做的事情就是重定义四条内部命令 \@oddhead, \@evenhead, \@oddfoot, \@evenfoot。我们把页设成空的，奇偶书眉一样。书眉是一个小页，用的是页面宽度，由一个表格组成，表格中就是相关的文档信息。

```
\newcommand{\ps@dochead}{%
  \renewcommand{\@oddhead}{%
    \begin{minipage}{\textwidth}\normalfont
      \begin{tabular*}{\textwidth}{@{}l@{\extracolsep{\fill}}}%
        \l@{\extracolsep{0pt}:~}l@{}}%
        \DH@number      & Version & \DH@version \\
        \DH@title       & Section & \thesection \\
        Date:~\DH@date  & Page    & \thepage
      \end{tabular*}\vspace{0.5ex} \rule{\textwidth}{0.6pt}%
    \end{minipage}}
  \renewcommand{\@evenhead}{\@oddhead}
  \renewcommand{\@oddfoot}{}
  \renewcommand{\@evenfoot}{}
}
\pagestyle{dochead}
```

最后那行就激活了新的页面样式。

还需要增加 \headheight 和 \headsep 的尺寸，因为我们这里的书眉要比通常的高很多。我们选取的高度是通常行距的 3.5 倍。

```
\setlength{\headheight}{3.5\baselineskip}
\setlength{\headsep}{3em}
```

我们下面进一步对上述定义进行修改。类似于上面这样的文档，它通常希望页码是在一节内部进行编号的。因此，我们要重定义 \thepage 命令，使其利用页码计数器的值显示页码 (7.1.4 节)，而且需要修改 \section 命令，使得每当调用它时，就开始一个新

页，并重设页码计数器的值。这要首先保存 `\section` 的当前定义，然后才进行重定义。

```
\let\DH@section=\section
\renewcommand{\thepage}{\thesection-\arabic{page}}
\renewcommand{\section}{\newpage\setcounter{page}{1}\DH@section}
```

注意这里新的 `\section` 命令要调用保存在内部命令 `\DH@section` 中的原来定义。

如果上述代码保存在一个叫 `dochead.cls` 的文件中，那么下面这样的主文件就可以调用它：

```
\documentclass{dochead}
\DocTitle{Spacecraft Cleanliness}
\DocNumber{ESA--XY--123}
\DocDate{2000 Feb 26}
\DocVersion{3.1}
\begin{document}
. . . . .
```

这样在第 3 节中第 4 页上的书眉就如下所示

ESA-XY-123	Version: 3.1
Spacecraft Cleanliness	Section: 3
Date: 2000 Feb 26	Page : 3-4

仿照这个例子，要为其他文档创建不同条目的书眉就应该不会是一件困难的事情。

C.3.3 为其他语言改编 L^AT_EX

标准 L^AT_EX 是基于英文而设计的，因此在很多地方会自动显示出英文单词，如 “Abstract” 或 “Contents”。虽然利用 L^AT_EX 可以生成有重音的字母，但是对于一长节文本而言，这样的输入方法是太复杂了。现在对于英文单词的断词模式也适用于带重音的字母。因此，如果要为其他语言改编 L^AT_EX，那么应该进行三方面的修改：

1. 用另外语言的翻译替换原来的英文单词，
2. 为重音或特殊标点符号定义简化的命令，
3. 改变断词式样。

如果所用的 T_EX 版本是比较新的，那么可以很好地做到第 1 点。起初，类似于 “Abstract” 和 “Contents” 这样的单词是显式地包含在生成这些章节的命令中。在欧洲，根据 Technical University of Vienna 的 H. Partl 的建议，来自于 Darmstadt 的 J. Schrod 建立了一个 L^AT_EX 修正版本，在这个版本中，上面所有英文单词都被类似于 `\abstractname` 和 `\contentsname` 这样的命令所代替。因此现在就非常容易进行相应于某种语言的修改，因为只要改变这些名称命令的定义即可，而不用重定义整个 `\abstract` 或 `\tableofcontents` 命令。这些名称命令从 1991 年 12 月 1 日开始已成为正式 L^AT_EX 的一部分。

相应于世界语的改编

我们选取一个示例, 来说明把 L^AT_EX 改编到国际语言——世界语中的原则。上面所列的三个问题都要加以考虑; 解决方法并不是显而易见的, 但相对于改编到法语和德语中而言, 复杂性则要低很多。

实际上, 已经有人设计了两个可用的世界语宏包, 一个叫 `espo.sty`, 它没有署名, 也没有日期, 功能也相当简单; 另一个叫 `esperant.sty`, 由德国 Mainz 大学的 Jörg Knappen 设计, 注明日期为 1991 年 12 月 10 日。后一个宏包的作者保证这个宏包以及 `german.sty` 的功能是稳定的。

世界语所用的字母表中有 28 个字母, 包括通常拉丁字母表中除 *q, w, x* 和 *y* 外的所有字母, 并增加了特殊字母 *ĉ, ĝ, ĥ, ĵ, ŝ* 和 *ŭ*, 分别对应于英语中的 *ch, j*, 德语中的 *ch*, 法语中的 *j*, 英语中的 *sh, w*。这些加了重音的字母看成是单独的字符, 并不像法语中认为是被装饰的字母, 或者德语中被作为其他字母组合的取代。抑扬重音在 L^AT_EX 中是用 `\`` 得到的, 而二全音符是用 `\u` 生成的。但这里要稍微复杂一些, 因为在 *h* 上的符号位置要有点儿改变, 而 *j* 的点要去掉。而且用 `\u{u}` 得到 *ŭ* 要输入五个键。为了给用户简化这种操作, `Esperanto` 样式生成了一个新的单字符命令 `^`, 对所有的特殊字母都有效。因此 `\`c \`g \`h \`j \`s \u` 的结果就是 *ĉ ĝ ĥ ĵ ŝ ŭ*。

然而, 要想得到上面那样的字符, 只是把 `^` 改变成主动(命令)字符, 并把它定义成 `\`` 是不够的。不但要把这条命令设成牢固的(2.7 节), 而且对于包含 `^` 的单词在断词的时候, 语言之间的切换, 以及最重要的利用名称命令对英文标题进行翻译的时候, 都有很多问题需要解决。

我们这里所给出的例子, 实际上是 Jörg Knappen 对 L^AT_EX 2_ε 进行改编而设计的宏包 `esperant.sty`。这个宏包解决了上面提出的所有问题。这个宏包原来必须在 Plain T_EX 下执行, 后来则为 L^AT_EX 2.09 的一个样式选项。它使用了相当多的 T_EX 技巧, 我们在此不做详细解释, 只说明一下最后的结果是什么。

做为一个宏包文件, 其开始也是鉴定语句:

```
\NeedsTeXFormat{LaTeX2e}[1994/06/01]
\ProvidesPackage{esperant}
[1994/06/08 1.1a2e Daly's adaption of Knappen's]
```

这是对 L^AT_EX 2_ε 的一个极小改编, 因此没有定义任何选项。我们就直接进入主体, 其中第一项利用一个 T_EX 技巧使得有重音的单词可以有断词。做法是加入一个空的单词断点, 但不允许断行。

```
\newcommand{\allowhyphens}{\penalty\@M \hskip\z@skip}
```

第二条语句把字符 `^` 加到某个特殊列表中, 而且必须把源文本原样输出。不必知道如何做到这点, 只要接受它就可以了。

```
\begingroup
\def\do{\noexpand\do\noexpand}%
\xdef\dospecials{\dospecials\do\`}%
```

```
\def\@makeother{\noexpand\@makeother\noexpand}%
\xdef\@sanitize{\@sanitize\@makeother\^}%
\endgroup
```

然后, 定义两条命令, 让抑扬字符 $\hat{\cdot}$ 的功能在做为上标运算符的普通角色与做为主动字符 (即单字符命令) 之间切换。

```
\newcommand{\modifiedhaton}{\catcode'\^ \active}
\newcommand{\modifiedhatoff}{\catcode'\^ 7 }
```

第三步, 定义抑扬命令, 然后把它变成主动字符, 并把它定义成有一个参数的牢固命令。在数学模式 (`\ifmmode`) 中就调用通常的上标函数 `\sp`, 否则就在处理之前检测参数值是否是特殊字符。如果参数是 `U` 和 `u`, 就显示二全音符重音; 如果是 `j`, 就用没有点的相应字符; 如果是 `h`, 就调用重叠函数。

```
\modifiedhaton
\DeclareRobustCommand{\hat}[1]{\ifmmode\sp{#1}
  \else\ifx#1u\allowhyphens
\else\ifx#1U\allowhyphens
  \else\ifx#1h\llap{\hat{}}\allowhyphens
  \else\ifx#1j{\hat{j}}\allowhyphens
  \else\ifx#1|\discretionary{-}{|}{|\allowhyphens
  \else{\hat{#1}}\allowhyphens
\fi\fi\fi\fi\fi\fi}
\modifiedhatoff
```

注意这里又为 $\hat{\cdot}$ 增加了一项功能: $\hat{|}$ 加入一个自由连字符, 即建议的单词断点, 利用这种方法, 单词余下的部分仍可以自动断词。通常在 \TeX 中, 单词断点只能出现在有自由连字符的地方。

第四步, 建立三条命令, 以定义世界语、美国英语和英国英语中的日期。在解释 `\hodiau` 和 `\hodiaun` 这两条命令时需要一点世界语的语法: 表示今天的单词是 *hodiaŭ*, 例如 *Hodiaŭ estas la 15a de novembro, 1994*, 当做为的一封信的日期时, 日期通常必须是宾格的 (*la 15an de novembro, 1994*)。我们对这两种可能都做了考虑。

我们不采用直接重定义 `\today` 的方法, 而是生成几个重定义的命令, 我们可以利用它们在世界语、美国英语和英国英语之间来回切换。后两者是用 `\providecommand` 定义的, 主要是为了防止已有宏包对它进行定义。

```
\newcommand{\dateesperanto}{\renewcommand{\today}
  {la \number\day -an de \ifcase\month\or
  januaro\or februaro\or marto\or aprilo\or majo\or junio\or
  julio\or a\u{u}gusto\or septembro\or oktobro\or novembro\or
  decembro\fi, \space \number\year}%
\let\hodiaun=\today}
\newcommand{\hodiau}{la \number\day -a de \ifcase\month\or
```

```

januaro\or februaro\or marto\or aprilo\or majo\or junio\or
julio\or a\u{u}gusto\or septembro\or oktobro\or novembro\or
decembro\fi, \space \number\year}

\providecommand{\dateUSenglish}{\renewcommand{\today}{%
\ifcase\month\or January\or February\or March\or April\or
May\or June\or July\or August\or September\or October\or
November\or December\fi \space \number\day, \number\year}}

\providecommand{\dateenglish}{\renewcommand{\today}{%
\ifcase\day\or 1st\or 2nd\or 3rd\or 4th\or 5th\or 6th\or 7th\or
8th\or 9th\or 10th\or 11th\or 12th\or 13th\or 14th\or 15th\or
16th\or 17th\or 18th\or 19th\or 20th\or 21th\or 22th\or
23th\or 24th\or 25th\or 26th\or 27th\or 28th\or 29th\or
30th\or 31st\fi~\ifcase\month\or
January\or February\or March\or April\or May\or June\or
July\or August\or September\or October\or November\or
December\fi \space \number\year}}

```

接下来要用 L^AT_EX(即 1991 年 12 月 1 日以后发行的 L^AT_EX 版本)中的名称命令对章节标题进行翻译。同前面处理日期时一样,我们也不是直接进行,而是用 `\captionesperanto` 和 `\captionseenglish` 进行我们的翻译工作。(注意,下面是把两组标题并列摆放,它们实际上是一个接在另一个后面的。)定义是用 T_EX 的 `\def` 命令进行的,因为它并不在意所定义的命令是否已经存在。

<code>\def\captionesperanto{%</code>	<code>\def\captionseenglish{%</code>
<code>\def\contentsname{Enhavo}%</code>	<code>\def\contentsname{Contents}%</code>
<code>\def\listfigurename{Listo</code>	<code>\def\listfigurename{List</code>
<code>de figuroj}%</code>	<code>of Figures}%</code>
<code>\def\listtablename{Listo</code>	<code>\def\listtablename{List</code>
<code>de tabeloj}%</code>	<code>of Tables}%</code>
<code>\def\abstractname{Resumo}%</code>	<code>\def\abstractname{Abstract}%</code>
<code>\def\partname{Parto}%</code>	<code>\def\partname{Part}%</code>
<code>\def\chaptername{\^Capitro}%</code>	<code>\def\chaptername{Chapter}%</code>
<code>\def\prefacename{%</code>	<code>\def\prefacename{Preface}%</code>
<code>Anta\u{u}parolo}%</code>	
<code>\def\appendixname{Apendico}%</code>	<code>\def\appendixname{Appendix}%</code>
<code>\def\refname{Cita\~{\j}oj}%</code>	<code>\def\refname{References}%</code>
<code>\def\bibname{Bibliografio}%</code>	<code>\def\bibname{Bibliography}%</code>
<code>\def\indexname{Indekso}%</code>	<code>\def\indexname{Index}%</code>
<code>\def\figurename{Figuro}%</code>	<code>\def\figurename{Figures}%</code>
<code>\def\tablename{Tabelo}%</code>	<code>\def\tablename{Table}%</code>

<code>\def\pagename{Pa\~go}%</code>	<code>\def\pagename{Page}%</code>
<code>\def\seename{vidu}%</code>	<code>\def\seename{see}%</code>
<code>\def\alsoname{vidu anka\u{u}}%</code>	<code>\def\alsoname{see also}%</code>
<code>\def\notesname{Rimarkoj}%</code>	<code>\def\notesname{Notes}%</code>
<code>\def\enclname{Aldono(j)}%</code>	<code>\def\enclname{encl}%</code>
<code>\def\ccname{Kopie al}%</code>	<code>\def\ccname{cc}%</code>
<code>\def\headtoname{Al}%</code>	<code>\def\headtoname{To}%</code>
<code>\def\subjectname{Temo}%</code>	<code>\def\subjectname{Subject}%</code>
<code>}</code>	<code>}</code>

这些名称只会出现特定的类中，而且有些也不是标准的名称。例如，`\refname` 只能用在 `article` 类中，在 `book` 和 `report` 类中其被 `\bibname` 所代替，而 `\abstractname` 也只会用在 `article` 和 `report` 类中。`\pagename`、`\enclname`、`\ccname` 和 `\headtoname` 只会出现 `letter` 类中，`\seename` 是为 `makeidx` 宏包准备的。其他命令都是非标准的，只用于特殊用途或改编，它们有：`\prefacename`、`\notesname`、`\alsoname`（为 `makeidx` 准备的）以及 `\subjectname`（为 `letter` 准备的）。

正如上面所指出的，在世界语中的重音字母是看成单独字符的。这也就是说，在某些情况下以字母为序进行排列时，它应显示为 ‘A B C Ĉ ...’ 或 ‘a b c ĉ ...’。为了做到这一点，要定义两个新的计数器类型：`\Esper` 和 `\esper`，其功能类似于 `\Alph` 和 `\alph`。

```
\newcommand{\esper}[1]{\@esper{\value{#1}}}  
\newcommand{\Esper}[1]{\@Esper{\vaule{#1}}}  
\newcommand{\@esper}[1]{\ifcase#1\or a\or b\or c\or \~c\or d\else  
  \@iesper{#1}\fi}  
\newcommand{\@iesper}[1]{\ifcase#1\or \or \or \or \or \or  
  \or e\or f\or g\or \~g\or h\or h\llap{\~{}}\or i\or j\or \~j\or  
  k\or l\or m\or n\or o\or p\or r\or s\or \~s\or t\or u\or  
  \u{u}\or v\or z\else\@ctrerr\fi}  
\newcommand{\@Esper}[1]{\ifcase#1\or A\or B\or C\or \~C\or D\else  
  \@Iesper{#1}\fi}  
\newcommand{\@Iesper}[1]{\ifcase#1\or \or \or \or \or \or \or E\or  
  F\or G\or \~G\or H\or \~H\or I\or J\or \~J\or K\or L\or M\or  
  N\or O\or P\or R\or S\or \~S\or T\or U\or \u{U}\or V\or  
  Z\else\@ctrerr\fi}
```

下面定义几个语言计数器以跟踪当前语言。它只对 3.0 版本以后的 TeX 才有效，这时把 `\language` 设置为一个数值，以激活存贮在该语言编号下面的断词式样。这里我们假定英语断词式样保存在零号语言中，而世界语式样保存在一号语言中。`\selectlanguage` 命令通过调用日期和标题定义命令来打开选定的语言。

```
\newcounter{USenglish} \setcounter{USenglish}{0}  
\newcounter{esperanto} \setcounter{esperanto}{1}
```

```

\newcounter{english}      \setcounter{english}{0}

\providecommand{\selectlanguage}{}
\renewcommand{\selectlanguage}[1]{%
  \ifcase \vaule{#1}%
    \dataUSenglish      \captionseenglish   \or
    \dateesperanto      \captionseesperanto \or
    \dateenglish        \captionseenglish   \fi
  \language=\value{#1}}

```

最后，定义切换世界语的开关，并用来在文档开头处打开世界语。

```

\newcommand{\EsperantoOff}{\modifiedhatoff}
\newcommand{\EsperantoOn}{\modifiedhaton}
\AtBeginDocument{\EsperantoOn\selectlanguage{esperanto}}

```

C.3.4 作者 – 年代引用

在 B.3.1 节我们指出标准的 L^AT_EX 不能处理引用是由作者和年代组成的参考文献样式，而只能处理引用是数字型的。我们在此给出一个宏包和参考文献样式，以解决这个问题。

宏包文件

我们给示例宏包命名为 `authyear`，这是现在使用非常广泛的 作者-年代 引用样式宏包 `natbib` (由 Patrick W. Daly 完成) 的一个非常简化的版本。但它确实演示了最重要的功能。

在 作者-年代 引用机制中，我们通常需要两种形式的引用，一种引用是放在括号内的，另一种则是平铺直述的。对这两种样式的区分是通过放在 `\cite` 命令中的可省注释参数值来实现的。假设在 `thebibliography` 中有如下一项条目：

```

\bibitem[{\it Jones et~al.}(1990)]{jone90}
  Jones, F. B., Smith, T. E., and Harris, R. E. ...

```

`jone90` 就是 `\cite` 命令和标签 (即 `{\it jones et~al.}(1990)`) 的关键词。对于这种情形，标签中年代两边的括号就相当于一个定界符，从而把作者与年代分开。然后如何处理它们，那就要看如何用 `\cite` 命令了：

<code>\cite{jone90}</code>	结果为	<i>Jones et al. [1990]</i>
<code>\cite[] {jone90}</code>	结果为	[<i>Jones et al., 1990</i>]
<code>\cite[page~30]{jone90}</code>	结果为	[<i>Jones et al., 1990, page 30</i>]

包围引用的括号可以是圆括号，也可以是方括号；在引用中的标点符号可以是逗号，也可以是分号。

宏包文件的开头仍是鉴别命令。它也需要 `ifthen` 宏包，因此就紧接着调用了该宏包。

```
\NeedsTeXFormat{LaTeX2e}[1995/06/01]
\ProvidesPackage{authyear}[1995/10/09 2.1 (PWD)]
```

```
\RequirePackage{ifthen}
```

下面就准备并执行选项。这些选项将设置各种命令以保存标点符号和括号类型。由于这些存贮命令并不是独有的内部命令，因此用户也可以在任何时候进行重定义。然而，标准选项应该可以满足通常的需要。首先定义存贮命令，然后用选项进行设置，并选定缺省选项 `round` 和 `colon`，最后就执行所选定的选项。

```
\newcommand{\citebegin}{} \newcommand{\citeend}{}
\newcommand{\citesep}{}
\newcommand{\auyrsep}{,}

\DeclareOption{round}{\renewcommand{\citebegin}{(}
                     \renewcommand{\citeend}{)}}
\DeclareOption{square}{\renewcommand{\citebegin}{[}
                       \renewcommand{\citeend}{]}}
\DeclareOption{comma}{\renewcommand{\citesep}{,}}
\DeclareOption{colon}{\renewcommand{\citesep}{;}}
\ExecuteOptions{round,colon}
\ProcessOptions
```

如果用下面定义的对内部命令 `\AY@cite` 进行了格式化，那么它就会显示出实际的引用。其第一个参数就是有格式的引用，第二个就是 `\cite` 的可省参数值，即可省的补注。如果没有给出这个补注，`#2` 就是句号（见下面的 `\cite`），引用显示时并不放在括号内（即平铺直述的引用）；否则，要把它以及补注包含在括号内（放在括号内的引用）。

```
\newcommand{\AY@cite}[2]
  {\ifthenelse{\equal{.}{#2}}
    {#1}
    {\citebegin#1%
     \ifthenelse{\equal{#2}{}}
       {}
       {, #2}%
     \citeend}}
```

实际上主要工作是由 `\cite` 命令完成的。这里所发生的事情是基于 $\text{\LaTeX}2_{\epsilon}$ 对 `\cite` 的定义。新的 `\cite` 命令有两个参数值，第一个就是可省的补注。然而，有补注就意味着要使用放在括号内的引用。补注实际上可以是空的，这得到一个放在括号内的没有补注的引用。为了做到这一点，（可省的）第一个参数值的缺省值为句号，它表示调用 `\cite`

时根本没有可省参数值，更不用说是一个空参数值了。

主要参数值 (#2) 就是一串引用关键词。它们必须一个一个地用 `\@for \@do` 进行处理。并把 `\@citeb` 相继设成列表中的关键词。`\@iden` 所在的行从 `\@citeb` 中去掉了前导空格；然后在辅助文件中为 Bib_TEX 生成一项条目；进行一次测试，看看关键词是否有相应的转换（保存在 `\b@\@citeb` 中），如果没有的话就显示出一条警告消息。此时，就有了新的功能：调用 `\AY@parse` 对关键词进行解释，把作者部分放到 `\AY@author` 中，年代部分放到 `\AY@year` 中；在平铺直叙引用 (#1=.) 中，要把这两部分放在一起，年代放在括号内。最后，调用引用显示命令 `\AY@cite`，第一个参数值为一串转换后的引用，第二个参数是 `\cite` 的可省参数值。注意在列表中每项引用前面加上了 `\AY@sep`；它起初是空的，但在第一项引用后变成 `\citesep`。这条命令可以在引用之间插入逗号或分号。

```
\newcommand{\AY@sep}{}

\renewcommand{\cite}[2][.]{%
  \renewcommand{\AY@sep}{}%
  \AY@cite{\@for \@citeb:=#2\do
    {\edef \@citeb{\expandafter \@iden \@citeb}%
     \ifthenelse{\boolean{@filesw}}
       {\immediate\write\@auxout{\string\citation{\@citeb}}}
     {}%
    \@ifundefined{b@\@citeb}
      {\AY@sep{\reset@font\bfseries ?}\G@refundefinedtrue
       \@latex@warning
        {Citation ‘\@citeb’ on page \thepage \space undefined}}
      {\AY@parse{\@citeb}%
       \ifthenelse{\equal{.}{#1}}
         {\AY@sep{\AY@author} \citebegin\AY@year
          \renewcommand{\AY@sep}{\citeend\citesep\ }}
         {\AY@sep{\AY@author}\auyrsep\ \AY@year
          \renewcommand{\AY@sep}{\citesep\ }}}}% ends \do
  \ifthenelse{\equal{.}{#1}}
    {\citeend}
    {}%
  }{#1}}
```

下面的命令对引用关键词转换后的结果进行解析，提取出作者和年代消息，它作为关键词调用时的参数。展开 (转换) 后的结果用做 `\AY@split` 的参数值。在此必须加上一个没有内容的 () 以防止转换后在括号内并不包含年代。`\let\protect=` 是为了得到牢固命令的特殊处理，用它是为了防止在此处就被展开，从而导致严重破坏。

```
\newcommand{\AY@parse}[1]{\let\protect=\@unexpandable@protect
```

```

\edef\@tempa{\@nameuse{b@#1}}%
\expandafter\AY@split\@tempa()\@}
\def\AY@split#1(#2)#3@{\gdef\AY@author{#1}\gdef\AY@year{#2}}

```

这就完成了对 `\cite` 及相关内部命令的重定义。我们现在来修补 `thebibliography` 环境。不幸的是，这意味着需要重定义整个环境，因为这里并不存在便利的内部命令可以修改。而且，我们需要判断是否处在 `article` 类中，即有没有 `\chapter` 命令，因为这确定了索引清单是否在 `\section*` 或 `\chapter*` 中。

```

\@ifundefined{chapter}
{
  \newcommand{\AY@bibsect}{\section*{\refname
    \mkboth{\MakeUppercase{\refname}}{\MakeUppercase{\refname}}}}
  \newcommand{\AY@bibsect}{\chapter*{\bibname
    \mkboth{\MakeUppercase{\bibname}}{\MakeUppercase{\bibname}}}}
\providecommand{\refname}{References}
\providecommand{\bibname}{Bibliography}

```

注意：`\MakeUppercase` 命令（以及这里没有用的 `\MakeLowercase` 命令）可以改变其参数值的大小写，相比于在 `LATEX 2.09` 中所用的 `TEX` 命令而言，这两条命令要更好一些。

我们现在来重定义 `thebibliography` 环境，使得标签并不会显示出来，第一行左对齐，所有后面的行向右缩进 1 cm。我们这里也要利用上面定义的 `\AY@bibsect` 命令。

```

\renewenvironment{thebibliography}[1]
{
  \AY@bibsect
  \setlength{\parindent}{0pt}\list{}
  {\setlength{\leftmargin}{1em}%
  \setlength{\itemindent}{-\leftmargin}}
  \ifthenelse{\boolean{@openbib}}
  {\renewcommand\newblock{\newline}}
  {\renewcommand\newblock{\hskip .11em plus.33em minus.07em}}
  \sloppy\clubpenalty4000\widowpenalty4000
  \frenchspacing}
{\renewcommand{\@noitemerr}{\@latex@warning
  {Empty 'thebibliography' environment}}}%
\endlist\vspace{-\lastskip}}

```

最后要做的一件事就是中止 `\bibitem` 命令的功能，不让它显示出标签。这里我们只需修改一条内部命令，并忽略其参数。

```

\renewcommand{\@biblabel}[1]{\hfill}

```

参考文献样式文件

在上面我们完成了 `authyear.sty` 宏包文件。然而，只有当 `thebibliography` 环境中的 `\bibitem` 命令具有本节开始所示形式时才有效。此时参考文献可以手工得到，也可以

利用 BibT_EX 程序,但这时必须存在一个正确格式的参考文献样式文件 (.bst)。虽然在标准 BibT_EX 宏包中没有这样的样式文件,但只要对 alpha.bst 文件稍做一点儿修改就可以得到。我们下面只是列出做到这点的必要代码,而并不详加解释,因为 BibT_EX 用的是与众不同的语言。

把 alpha.bst 文件复制到一个名为 authyear.bst 的新文件中,然后在其中找到文本 FUNCTION {output.bibitem}。这个函数的第五行是字符串 "]{ " write\$; 把它修改成 ")]{" write\$, 即在右方括号前面加上右小括号。

然后,找到 FUNCTION {format.date}, 并在这个函数最后那个右大括号前面加上字符串 extra.label *, 即现在为

```
if$
  extra.label *
}
```

接着找到文本 FUNCTION {format.lab.name}, 并用下面的代码替换这个函数的所有文本(从开头到 FUNCTION {author.key.label} 之前的内容):

```
FUNCTION {format.lab.names}
{ 's :=
  s #1 "{vv~}{ll}" format.name$
  s num.names$ duplicate$
  #2 >
    { pop$ " et~al." * }
    { #2 <
      'skip$
      { s #2 "{ff }{vv }{ll}{ jj}" format.name$ "others" =
        { " et~al." * }
        { " and " * s #2 "{vv~}{ll}" format.name$ * }
        if$
      }
      if$
    }
    if$
  }
}
```

最后,查找 FUNCTION {calc.label}, 并定位到包含 duplicate\$ 的那行。在其后加上字符串 emphasize "(" *. 在接下来的一行中,把 #2 改成 #4。因此这些行及相邻行现在应该为

```
if$
duplicate$
emphasize "(" *
year field.or.null purify$ #-1 #4 substring$
```

*

'label :=

(上面的 `emphasize` 指的是要用斜体显示引用中的作者，而不是索引中的作者。如果不希望这样，那就去掉它。)

注意：这个新参考文献样式只在 0.99 或以后版本的 BibTeX 中才可以用。

为了演示如何使用这一样式，假定在参考文献数据文件 `mylit.bib` (见 B.2 节) 中有一项条目：

```
@ARTICLE{jone90,
  AUTHOR = {Fred B. Jones and
            Thomas Elwood Smith and
            Richard E. Harris},
  TITLE = {Activity in the Night-Side Ionosphere},
  YEAR = {1990},
  JOURNAL = {J. Geophys. Res.},
  VOLUME = {101},
  PAGES = {1234-1254} }
```

如果在 LaTeX 文件中现在包含如下文本行

```
\documentclass{article}
\usepackage[square]{authyear}
\begin{document}
... as shown by \cite{jone90}, it is possible ...
... has been measured \cite[]{jone90}. This is ...
\bibliographystyle{authyear}
\bibliography{mylit}
\end{document}
```

所得的输出会是

```
... as shown by Jones et al. [1990], it is possible ...
... has been measured [Jones et al., 1990]. This is ...
```

Reference

Fred B. Jones, Thomas Elwood Smith, and Richard E. Harris. Activity in the night-side ionosphere. *J. Geophys. Res.*, 101:1234–1254, 1990.

附录 D 错误与警告消息的处理

在编写长 L^AT_EX 文档时很容易出现错误。这里的错误可以是各种类型的，从最简单的命令名称输入错误到忘记了某些命令必须配对，或者忘记了复杂命令的语法等等种类的错误都有可能发生。

在 L^AT_EX 处理过程中的错误会在屏幕上显示出一长串消息，这些信息对初学者而言，是完全不可理解的。即便是高级用户在确定出错的具体地方时也会有一些困难。然而，这些消息包含了导致问题的基本结构信息，可以帮助有经验的 T_EX 使用者深入了解问题本质。

另外，错误消息中也包含对初学者而言相当有用的信息。本章的目的就是解释一下这些对非程序设计者有帮助的消息。

D.1 错误消息的基本结构

错误消息有两个来源：一些来自于 L^AT_EX，另一些来自于基本的 T_EX 程序。由于 L^AT_EX 是在更高层次上进行操作，因此 L^AT_EX 消息之后通常接着 T_EX 消息。

D.1.1 T_EX 错误消息

我们从一个简单的错误例子开始。

```
\documentclass{article}
\begin{document}
The last words appear in \txetbf{bold face}.
\end{document}
```

其中 `\textbf` 命令被误输入为 `\txetbf`。在处理过程中，L^AT_EX 认为作者想调用的是 T_EX 命令 `\txetbf`。由于 L^AT_EX 不认识 T_EX 命令，它只是把这条命令送给 T_EX 处理，在这里可以确定其指令系统中并没有这条命令。那么就会在屏幕上显示出下面的错误消息：

```
! Undefined control sequence.
1.3 The last words appear in \txetbf
                                {bold face}.
?
```

程序运行到此处就停下来，等待用户给出反馈。这条消息即使是初学者也能理解。其中包含由惊叹号！开始的错误标识。这里的标识为 `! Undefined control sequence`，指的是未知的命令名称（控制序列）导致出错。接下来是两行文本，第一行前缀 1.3，指的是错误出现在输入文本的第 3 行。而错误本身就是在这一行最后那个符号中遇到的。而下面一行显示的是当遇到这个错误时正在处理的输入行的其他部分，这里是单词 `{bold face}`。消息中最后一行的问号标志表示 T_EX 需要用户的反应。

在这里输入一个？，并回车，那么就会显示出如下信息：

```
Type <return> to proceed, S to scroll future error messages,
R to run without stopping, Q to run quietly,
I to insert something, E to edit your file,
1 or ... or 9 to ignore next 1 to 9 tokens of input,
H for help, X to quit
?
```

下面就是用户可能给出的反应清单：

1. 〈回车〉：简单地输入回车键，让 \TeX 在经过一番按照预先设计好的规则，处理这个错误后，继续向下处理。在这种出现未知命令名称的情形中，处理错误的方式就是这样来忽略它。
2. S 滚动模式： \TeX 继续向下处理，当再次遇到错误时，还会在屏幕上显示出消息，但并不等用户做出反应。这就好像在所有后续错误中按的都是〈回车〉键。
3. R 运行模式： \TeX 如同 S 一样继续向下处理，但是即使遇到类似于在 `\input` 或 `\include` 命令中没有文件名做参数这样的错误也不停下来。
4. Q 安静模式：同 R 一样，只是不会再向屏幕上显示错误消息。然而这些消息要写到 `.log` 文件中。
5. I 插入：通过插入正确的文本来校正错误。 \TeX 把从键盘上输入的文本行取代出错的地方，然后继续处理下去。对于这样的校正，实际上 `.tex` 文件中的原始文本并没有改变，还是需要用编辑器进行修改。输入 `\stop` 会使得在 `.dvi` 文件的当前页上终止程序。
6. 1 ...：输入一个小于 100 的数，从而在后续文本中删掉相应数目的字符。程序还是会在新地方停下来等用户的反应。
7. H 帮助：在屏幕上显示出对问题的进一步考虑，它由相对于简短错误标志要多很多的信息组成，从而可能包含改正错误的有用提示。
8. X 退出：在该点停止执行 \TeX 处理过程。当前页并不出现在 `.dvi` 文件中。
9. E 编辑：同 X 一样，处理过程中止，并显示信息，说哪个文件的哪一行出现了错误。在有些实现版本中，可能自动调用编辑器，而且直接跳到出错行。

上面的反应字母既可以是大写的，也可以是小写的。只有按了〈回车〉键，反应才有作用。

在前面那个示例错误消息中，若按 H 或 h 以寻求帮助，会得到如下文本：

```
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., '\hobx'), type 'I' and the correct
spelling (e.g., 'I\hbox'). Otherwise just continue
and I'll forget about whatever was undefined.
```

？

这里更详细地描述了错误情形：在上面一行尾部的命令名称是未知的；如果这只是一个输入错误，那么就用 I 输入正确的文本，这里是 `I\textbf`。否则，按 (回车) 键，有错的命令被忽略。对这行的处理就如同是 `The last word appears in bold face`。当然，这里并不会真的得到黑体。

\LaTeX 的错误消息结构总结如下：

每条错误消息都以错误指示开始，其第一行的开头有 ! 号。所谓指示文本就是对问题的简要说明。接下来是一行或多行的输入文本。其中第一行的最后那个符号使得 \TeX 停下来，并显示错误消息。最后一行由接下来要处理的文本或命令组成。 \TeX 要等待用户的反应。如果这里的反应是寻求帮助 H，那么就会在屏幕上显示出更详细的说明，这可能会给出一些提示，而且 \TeX 还会等待进一步的反应。

D.1.2 \LaTeX 的错误消息

在 $\text{\LaTeX}2.09$ 与 $\text{\LaTeX}2_{\epsilon}$ 之间的一个主要差别就是它们的错误消息样子不同。在原来的版本中会显示出许多行难以理解的文本，以揭露出导致出错的深层内部代码，而新的版本只是显示出错误大概发生在哪几行文本中。在下面一节我们会讲到来自于 $\text{\LaTeX}2.09$ 的错误消息例子，除此之外所有其他的例子都是针对于 $\text{\LaTeX}2_{\epsilon}$ 的，发行日期为 <1994/06/01>。

为了给出一个有 \LaTeX 错误的文本例子，我们如下输入：

```
\documentclass{article}
\begin{document}
\begin{quote}\slshape
  Text indented at both ends
\end{quote}
\end{document}
```

这里 `\begin{quote}` 调用中错误输入为 `quote`。当 \LaTeX 处理这段文本时会显示如下错误消息：

```
! LaTeX Error: Environment quote undefined
```

```
See the LaTeX manual or LaTeX Companion for explanation.
```

```
Type H <return> for immediate help.
```

```
...
```

```
1.3 \begin{quote}
```

```
      \slshape
```

```
?
```

这段消息的第一行就有一个简短的错误指示，说明这是 \LaTeX 自己发现的一个错误，这里是 `Environment quote undefined`。所有的 \LaTeX 错误都是以类似这样的一行开始的，接着告诉若想得到详细解释，请看 \LaTeX 手册（在本书的 D.3 节也可以看到这一介绍）。

文本的第三行提示利用反应 H(回车) 可以得到其他的说明。

有三个点 ... 的那一行表示这里没有写出来的有关内部代码的许多行。在 \LaTeX 2.09 中会显示出这些内部代码行, 即使在 \LaTeX 2_ε 中, 也可能有选择地显示出一些。请见 D.1.3 节。

接下来两行显示的是当发现错误时处理过程所停住的地方。同 \TeX 消息一样, 文本的当前输入行也是断在出错的地方, 前面部分放在第一行上, 这里的断点为 `\begin{quote}`, 其他部分放在下面的行中。行指示符 1.3(小写字母 L, 不是数字 1) 表示是在输入文件的第 3 行上遇到错误的。

\LaTeX 现在等用户给出一个反应。输入 H(回车) 会得到额外的信息:

```
Your command was ignored.
```

```
Type I <command> <return> to replace it with another command,  
or <return> to continue without it.
```

```
?
```

这就是说以 1.3 开始的两行中上面这行的最后那条命令还没有进入处理过程中。因此可以用只适用于当前处理过程的 `I\begin{quote}`(回车) 方法来校正。但是这里文件中的拼写错误仍然没改正过来, 需要稍后再运行编辑器把它改过来。

然而, 如果只是简单地输入 (回车) 键, 处理过程就会继续下去, 这样就会忽略出错的 `\begin{quote}`, 好像它在文件中不存在一样。可是当遇到 `\end{quote}` 命令时这又会直接导致另一个错误, 因为这里没有配对的 `\begin{quote}` 命令。这第二条错误消息的内容为:

```
! LaTeX Error: \begin{document} ended by \end{quote}
```

```
See the LaTeX manual or LaTeX Companion for explanation.
```

```
Type H <return> for immediate help.
```

```
...
```

```
1.5 \end{quote}
```

```
?
```

这条消息中照样还是包含标准的 \LaTeX 声明, 即有错误指示的第一行, 接下来提醒参考手册, 以及输入 H 可以得到帮助。在这种情形中的错误指示为

```
\begin{document} ended by \end{quote}
```

这是因为当 \LaTeX 遇到 `\end{quote}` 时, 它会检查当前环境的名称。由于前面没有相应的 `\begin{quote}`, 因此匹配的 `\begin` 命令就是 `\begin{document}` 声明, 从而得到一个匹配错误的 `\begin ... \end`。

最后两行显示的仍然是当发现错误时, 处理已进行到了哪里。这里整个当前行都被考虑进来了, 因此另一行是空的。

在这时的反应 H(回车) 会得到与第一个错误相同的消息。这里利用 I 进行改正并不会取得好效果, 因为这里再不可能把不存在的 `\begin{quote}` 插入到环境文本的前面。输入 `I\begin{quote}` 只是用它代替 `\end{quote}`, 这并没有解决实际问题。现在最好的反应就是输入 (回车), 这样 `\end{quote}` 命令也被忽略, 处理过程继续下去。

这样到现在为止, 有错的 `\begin{quote}` 和正确的 `\end{quote}` 命令都被去掉了, 处理过程就如同在源文件中没有用 `quote` 环境一样。

如果是在 `\end` 处犯同样的拼写错误, 而在 `\begin` 处则拼写正确, 那么会得到如下错误消息:

```
! LaTeX Error: \begin{quote} on input line 3 ended by \end{quote}
```

```
See the LaTeX manual or LaTeX companion for explanation.
```

```
Type H <return> for immediate help.
```

```
...
```

```
1.5 \end{quote}
```

```
?
```

前面的解释对读者理解这条消息的内容应该是足够了。这里的错误指示为

```
\begin{quote} on input line 3 ended by \end{quote}
```

最后两行文本表明问题出现在第 5 行上, 导致麻烦的命令是 `\end{quote}`。现在明显可以采取的方法就是用 `I\end{quote}` 进行改正, H 消息也支持采取这一操作。然而, 现在会在屏幕上显示出新的错误消息:

```
! Extra \endgroup
```

```
<recently read. \endgroup
```

```
1.5 \end{quote}
```

```
?
```

这里除了以 1.5 开始的最后两行外, 其他的都好像一点儿道理也没有。错误指示 `! Extra \endgroup` 好像没有任何意义。这是一个 \TeX 错误, 而不是 \LaTeX 错误, 它对我们没有任何帮助。

此时不应责备受了挫折的用户。这里的反应是相当合理的, 虽然它还是错的。只有在具备了丰富的经验后, 我们才会知道此时最好的方法就是按 (回车)。这样就会关掉 `quote` 环境, 当然与 `\end{quote}` 相联系的任何特殊操作也随之去掉了。

此时的帮助消息是:

```
Things are pretty mixed up, but I think the worst is over.
```

这一消息至少是一个鼓励, 读者不要灰心。最好的方法就是继续按 (回车), 以结束这次运行。

这里我们给出如何选择反应的两条建议，一条非常具体，另一条很一般。具体的这条是：

如果出错的环境名位于 `\begin` 命令中，正确的改正错误的方法是

`I\begin{正确的名称}`

如果拼写错误出现在 `\end` 命令中，那么最好的处理方法就是按〈回车〉。这样就会关闭这个环境，任何局部声明或定义也会终止作用。然而，如果 `\end` 命令执行了某命令，或显示出一些文本，那么这些结果也同时消失了。

一般性建议是：

如果借助于错误消息，用户知道了如何改正错误，那么可以用

`I 修正的文本`

来进行。否则，用户可以按〈回车〉，等等看会出现什么结果。即使出现更古怪的 (T_EX) 错误，用户也可以持续按〈回车〉键，直到处理过程最终结束。那么接下来的输出结果会指出错误原因。

当然也可以不按〈回车〉键，而是先输入 S, R 或 Q，再按〈回车〉，以加速对错误的处理 (D.1.1 节)。在这里，如果只是按〈回车〉，错误的命令并没有被忽略。T_EX 在尝试猜测用户此处想进行的操作，从而进行一些改正。只有这一点行不通时，T_EX 才会完全忽略这条命令。例如，如果错误指示为

`\begin{环境} ended by \end{环境}`

这里至少有一条 `\begin` 命令中有一个非法的环境名称。这样就可以假设在 `\end` 命令中的环境名也是错误的。L^AT_EX 就会尝试利用当前环境的名称来执行这条命令。

D.1.3 来自于 L^AT_EX 2.09 中的错误消息

为了进行对比，我们先把上小节例子中的 `\documentclass` 换为 `\documentstyle`，然后我们看看来自于 L^AT_EX 2.09 中同样错误的消息结构。第一行消息显示为：

`! LaTeX Error. See LaTeX manual for explanation.`

`Type H <return> for immediate help.`

`! Environment qoute undefined.`

`\@latexerr ...diate help.}\errmessage {#1}`

`1.3 \begin{qoute}`

?

这里主要的差别就是错误指示现在位于第三行上，而不是作为第一行上 L^AT_EX 错误声明的一部分。另外一个差别就是第四行上的古怪文本，开头为 `\@latexerr ...`，这实际上是生成这个消息的内部代码，只要简单地忽略它就可以了。

如果不做任何修改，只是按〈回车〉键，那么会得到第二条错误消息，它与 L^AT_EX 2_ε 中的样子极端不一样：

```

LaTeX error. See LaTeX manual for explanation.
      Type H <return> for immediate help.
! \begin{document} ended by \end{quote}.
\@latexerr ...diate help.}\errmessage {#1}

\@checkend ...urrenvir \else \@badend {#1}
                                \fi
\end ...me end#1\endcsname \@checkend {#1}
                                \expandafter ...

1.5 \end{quote}

```

?

在 `\@latexerr...` 和 `1.5...` 之间多出来的那些行显示的是 `\end` 命令的内部代码, 通常 \LaTeX 用户对这些内容不会有任何兴趣。它们只会令人感到混乱。如果你仍然用的是 $\text{\LaTeX}2.09$, 而且遇到了这样的错误消息, 那么你就应该把这些行都忽略过去。除此之外, 错误指示、用 `1.n` 开始的输入行、帮助消息和需要的反应都与 $\text{\LaTeX}2_{\epsilon}$ 中完全一样。

有时候需要深入地观察错误消息, 这一点对那些知道这些消息意味着什么的有经验 \TeX 专家特别有用。在这种情况下, 可以利用下面的指令在 $\text{\LaTeX}2_{\epsilon}$ 中补上被去掉的代码行:

```
\setcounter{errorcontextlines}{数}
```

这里的 `数` 就是在错误之处宏被解码的深度层数。在默认情形中, $\text{\LaTeX}2_{\epsilon}$ 取 `数 = -1`, $\text{\LaTeX}2.09$ 取 `数 = 5`。有经验的用户可以把它设为 5 或其他的值以获得额外的信息。

D.1.4 来自于 \TeX 宏的错误消息

绝大多数的 \TeX 命令和实际用的 \LaTeX 命令都可以称为 \TeX 宏。它们就是原语命令的组合, 从而具有一个新的命令名称, 这样就可以整体调用它们。 \TeX 宏在结构上类似于 \LaTeX 中用 `\newcommand` 命令定义的对象。可以向它传递多达 9 个的参数值, 这也与 \LaTeX 命令一样。然而, 生成宏的相应 \TeX 命令要比 `\newcommand` 更具一般性。

事实上, 大约有 900 条 Plain \TeX 命令可以使用, 其中只有 300 条是原语, 或者称为基本命令。其余 600 条都是宏。如果在宏中出现了错误, 那么它里面的其他命令也可能受牵连。

为了明白起见, 这里给出一个例子。命令 `\centerline` 就是如下定义的宏:

```
\def\centerline#1{\@oline{\hss#1\hss}}
```

这里 `\@oline` 本身还是一个宏, 而 `\hss` 为一个 \TeX 原语, 它是一个弹性长度, 可以无限伸展或收缩。为了避免误导读者探讨太深层的 \TeX 命令, 这里我们只是指出上面定义的宏就基本上等价于 \LaTeX 命令序列:

```
\newcommand{\centerline}[1]{\makebox[\textwidth][c]{#1}}
```

下面给出示例文本:

```
\documentclass{article}
\begin{document}
\centerline{This is an \invalid command}
\end{document}
```

其中由于在单词 `invalid` 前面加了一个 `\`，从而生成一条错误命令 `\invalid`。在 \LaTeX 处理它时，会给出如下 \TeX 错误消息：

```
! Undefined control sequence.
<argument> This is an \invalid
                        command
1.3 ...erline{This is an \invalid command}
```

?

这条消息现在应该是很容易理解的。错误指示与 D.1.1 节例子中的相同：

```
! Undefined control sequence.
```

接下来两行文本说明是在处理完 `\invalid`“命令”时发现错误的，在它的后面将要读入的文本是单词 `command`。同时，在上面那行开头部分的 `<argument>` 说明这部分文本是某个命令的参数值。

再下面两行文本与前面两行类似，即错误出现在输入文本的第 3 行，而且整行文本（包括命令和参数值）都已经被读入并进行了处理。

D.2 一些错误样例

D.2.1 错误的传播

我们在前面给出了包含 `\begin{quote}` 环境的错误示例，这个例子表明如果只是给出一个简单地反应（回车），尽管 `\end{quote}` 命令是正确的，也会导致第二个错误消息。在实际处理过程中，会经常出现这种由于不正确的纠正而导致进一步错误的现象。

现在看一些源文本：

```
\documentclass{article}
\begin{document}
\begin{itemie}
  \item This is the first point in the list
  \item And here comes the second
\end{itemize}
\end{document}
```

这部分文本中的唯一错误就是把环境名 `itemize` 错写为 `itemie`。在 \LaTeX 处理时，首先生成与前面不正确的 `quote` 环境中相同的错误消息：

```
! LaTeX Error: Environment itemie undefined
```

See the LaTeX manual or LaTeX Companion for explanation.

Type H <return> for immediate help.

...

1.3 \begin{itemize}

\slshape

?

这时候, 如果用户反应为 I \begin{itemize}, 那么就会纠正这个错误, 从而顺利地结束处理过程。然而, 如果输入的只是 <回车>, 那么就会得到一条新的错误消息:

! LaTeX Error: Lonely \item--perhaps a missing list environment.

See the LaTeX manual or LaTeX Companion for explanation.

Type H <return> for immediate help.

...

1.4 \item T

this is the first point in the list

?

之所以会出现这样的结果, 是因为 \begin{} 命令被忽略, L^AT_EX 就认为是在列表环境外面使用 \item 命令, 这样 \item 没有任何意义。(实际上, 它在这里是有意义的: 它显示出了一条错误消息!) 现在要插入 itemize 环境的开始部分就太晚了。输入 H(<回车>) 可以得到如下帮助:

Try typing <return> to proceed.

If that doesn't work, type X <return> to quit.

?

遵照这条建议, 按了 <回车> 键, 那么就会又得到同样的错误消息, 但这次是出现在第 5 行上, 相应于第二条 \item 命令。继续按 <回车>, 就会得到:

! LaTeX Error: \begin{document} ended by \end{itemize}

See the LaTeX manual or LaTeX Companion for explanation.

Type H <return> for immediate help.

...

1.6 \end{itemize}

?

现在 itemize 环境总算结束了, 但是因为其开头不正确, L^AT_EX 抱怨遇到了不匹配的 \begin 和 \end 命令。在这里最后一次按 <回车>, 就会使处理继续进行下去。当然列表

环境中的内容是不会有正确的格式的，但是其他部分的文档不会受到影响。

在这个例子中，源文本的一个错误生成了三个其他错误消息，这是很常见的现象。有些 \LaTeX 错误可以导致上百条后续错误，甚至有可能错误链永不会中止，处理过程不再向前进展。在这种情况下，没有别的办法了，只能终止程序。为此可以通过在错误消息后面输入反应 \backslash stop 来做到。有时可能需要给出几次这种输入，其才会发生作用。如果这还行不通，也就是说每次还是出现同样的错误消息，那么用反应 \X (回车) 就可以马上结束程序的运行。

用 \backslash stop 要比用 \X 结束程序好，因为这样在输出中会包含最后一页的结果。这对于要推断错误来源时是非常有用的。

本节最后要告诉您的一条经验就是：即使遇到成群的错误，也不要惊慌！坚持按 (回车) 键，继续处理下去。

如果按的是 S (回车) 会在屏幕上得到同样的错误消息，但是其间不会再停下来等待用户的反应 (D.1.1 节)。

D.2.2 典型的严重错误

有的时候，用户可能忘记了 \documentclass 或 $\text{\begin{document}}$ 命令，甚至忘记提供所有的导言部分，这时就会导致严重的错误。比如说，若一个 \LaTeX 文件本来需要用 \input 或 \include 命令读入，但却直接用 \LaTeX 程序处理它，那么就会出现没有导言的错误。如果一个文件中的文本为

```
This file has no preamble.
```

这时若直接用 \LaTeX 进行处理，那么就会在屏幕上显示如下错误消息：

```
! LaTeX Error: Missing \begin{document}.
```

```
See the LaTeX manual or LaTeX Companion for explanation.
```

```
Type H <return> for immediate help.
```

```
...
```

```
1.1 T
```

```
his file have no preamble.
```

```
?
```

在输入 H (回车) 后得到的相应帮助消息为：

```
You're in trouble here. Try typing <return> to proceed.
```

```
If that doesn't work, type X <return> to quit.
```

```
?
```

从这条错误消息我们知道 \LaTeX 在读入第一个字母的时候就发现了一个错误。这里的帮助消息也不能提供更多的信息。在这里持续按 (回车) 是没有什么用的。反而我们应该用 \X 或 \E 马上停下来，因为继续处理下去，也得不到任何结果。

即使上面的例子文件中包含如下环境：

```
\begin{document}
```

```
  This file has no preamble.
```

```
\end{document}
```

也不可能得到一个正确的处理。只不过现在的 TeX 错误消息为:

```
! LaTeX Error: The font size command \normalsize is not defined:
there is probably something wrong with the class file.
```

See the LaTeX manual or LaTeX Companion for explanation.

Type H <return> for immediate help.

...

1.1 \begin{document}

?

这里的错误指示相当古怪,因为在输入中从没有用 \normalsize。只有消息中的最后两行还好理解。它们说明是在读入第 1 行的 \begin{document} 命令后发现错误的。实际上从文本的第一行就可以知道这个处理不会得到任何结果,因此在它前面并不存在不可缺少的 \documentclass 命令。

这时候,没有别的选择,就只能用 X 或 E 终止程序,因为继续处理下去也没有任何意义。

之所以出现如此奇怪的错误指示,是因为许多格式化参数需要用 \begin{document} 声明进行初始化,其中就包含对标准字体的定义。由于这里是内部调用了 \normalfont,而这条命令必须在类文件中进行定义,但这里并不存在类文件,所以出现这种错误提示。而其他的初始化命令是在 L^AT_EX 格式中定义的,因此它们不会导致错误。

如果文档类的名称输入错误,例如:

```
\documentclass{artice1}
```

那么就会得到如下错误消息:

```
! LaTeX Error: File 'artice1.cls' not found.
```

```
Type X to quit or <RETURN> to proceed,
or enter new name. (Default extension: cls)
```

Enter file name:

这条消息应该是相当明白的:程序正在寻找一个叫 artice1.cls 文件,但没有找到,因此它希望用户输入另外一个名称。如果新的文件具有所要求的扩展名 (.cls),那么就不需要显式给出。在当前情形下,任何标准 L^AT_EX 类的名称,如 article, report, book 或 letter,都可以作为输入,当然也可以输入其他可用类的名称。但是文档应该是相应于该类而编写的。

只要系统找不到要读入的文件，就会显示同样的错误消息。会导致这种问题的命令有 `\input`, `\include` 或 `\usepackage`。如果这个文件确实不存在，那么就可能是因为它并不在 \TeX 查找文件的路径中。通常在安装时要建立一个系统参数，告诉 \TeX 在哪些路径中查找文件。如果文件放在其他目录中，那就应该输入完整的文件名，即包含目录或路径标识符。

如果 \LaTeX 要求输入的文件并不存在，或者是您根本就不知道的文件，那您就应该马上终止处理或者告诉 \LaTeX 忽略这个烦人的文件。这时，就会遇到另一个问题，那就是 \LaTeX 坚持要求得到一个合法的文件名。输入 \X (回车) 只会得到同样的消息，这时它会说它找不到文件 `x.ext` (`ext` 指相应的文件扩展名)。有些安装版本中提供了一个没有任何内容的文件，名为 `null.tex`，因此这时可以输入 `null.tex`。另外还可以使用一个扩展宏包 (8.8.4 节的 `fileerr.dtx`)，它提供了一组文件，依据标准反应字母而命名为 `x.tex`, `e.tex`, `r.tex`, `h.tex`, `s.tex`，这样来模拟处理普通错误消息时的反应。

紧急停止：有时候我们会遇到一种情形，即使用 \I\stop 或 \X 也不能在出现错误后令程序终止。这时，就必须借助于操作系统的程序中断了。细节请看计算机中心或 PC 机的手册。

D.2.3 数学错误

对数学公式命令的不正确应用，会导致多得令人吃惊的错误，这一点对没有经验的用户更是如此。在数学公式中的经常见到的错误是一些无意识的错误，如忘记了一个右大括号 `}`，或者没有及时切换回文本模式。另外一种常见的错误是在文本模式中用了只能出现在数学模式中的符号。下面我们给出几个典型的例子。

如果我们想得到：‘The price is \$3.50 and the order number is type_sample’，那么可能会输入如下源文本：

```
The price is $3.50 and the order number is type_sample.
```

这块文本中包含两处错误，而且第一个错误取消了第二个错误：`$` 符号是在文本中生成数学公式的数学模式切换开关 (5.1 节)。在文本中生成真正的美金符号是输入 `\$`。然而，在这段示例文本中，单独的 `$` 是切换进入数学模式，把其后的内容做为公式处理。然而，这里没有关闭的切换开关 `$`，当这段结束时 \TeX 才会注意到这一点。

```
! Missing $ inderted.
```

```
<inserted text>
```

```
$
```

```
1.5
```

```
?
```

如果只是用 \X 做为反应， \TeX 会在该点插入一个 `$`。对于我们的示例文本，这就是在空行前面的文本结尾时插入 `$`，也就是说从第一个 `$` 到这段结束之间的文本都被当做正文公式处理了：

```
The price is 3.50andtheordernumberistype,$ample.
```

从这个结果马上就可以看出用户犯的错误了。在靠近行的尾部，有一个字母 `s` 显示为下标了。这就是该例中的第二个错误。下划线字符 `_` 只能用在数学模式中，这里应该输入为 `_`。然而，由于第一个 `$` 已（不正确地）切换进入数学模式，`_` 符号成为合法的了，这样其后字母 `s` 就成为下标了。

如果现在在文本中用 `\$` 代替 `$`，那么 `_` 符号就不是处于数学模式中，会生成如下错误消息：

```
! Missing $ inserted.
```

```
<inserted text>
```

```
$
```

```
1.5 ...$3.50 and the order number is type_
```

```
sample.
```

```
?
```

在这里按〈回车〉，告诉 $\text{T}_{\text{E}}\text{X}$ 在数学命令 `_` 前面插入 `$`，从而改正这个错误。这样处理过程就会继续下去，当到了当前段结束前的某处， $\text{T}_{\text{E}}\text{X}$ 就会注意到并没有闭 `$` 符号。这样就会显示与前面那种情形相同的错误消息。再按一次〈回车〉键，处理就会继续，得到如下结果：

```
The price is $3.50 and the order number is typesample.
```

在所有这三种情形中，都可以用 `H` 寻求更多的帮助，内容为

```
I've inserted a begin-math/end-math symbol since I think
you left one out. Proceed, with fingers crossed.
```

上面最后一句话就是我们所推荐的处理数学错误的方法：持续按〈回车〉或者 `S`，以使处理完成，这时再看打印的结果，以便找到错误。

D.2.4 来自多文件的错误

如果文档被分成许多文件，每个用 `\input` 或者 `\include` 命令读入，那么错误消息中的行号相应于正在读的文件。利用反应 `E`〈回车〉，将会调用编辑器程序，并打开文件，跳到发现错误的指定行。利用其他反应，就必须单独使用编辑器，而且显式地给定出错文件名称。

通过查看处理消息或者 `.log` 文件，可以知道出错时正在处理的是哪个文件。当文本被打开供读取时， $\text{T}_{\text{E}}\text{X}$ 会在屏幕或 `.log` 文件中写一个左小括号（以及文件的名称。当文件被关闭时，会写右小括号）。输出页码还是像通常那样显示在中括号内，这些内容都是同时显示在屏幕和写进 `.log` 文件中的。例如，如果屏幕上如下处理消息：

```
..(sumfile.tex [1] [2] [3] (part1.tex [4] [5]) (part2.tex [6] [7]
```

```
! Undefined control sequence
```

```
1.999 \finish
```

```
?
```

那么可以有如下解释：一个叫 `sumfile.tex` 的文件正在被读取，而且在输出完第 1,2,3 页后，文本 `part1.tex` 被读取（在文件 `sumfile` 中用 `\input` 和 `\include` 命令），这时输出

第 4,5 页, 然后关闭 `part1.tex`, 接着打开 `part2.tex` 文件做输入。在这个文件的第 999 行上发现一个错误。如果从键盘上改正了这个错误, 或者如果处理过程还是继续下去了, 屏幕上会有如下消息:

```
[8] [9]) [10]
! Too many }'s
1.217 \em sample}
```

在第 9 页后面的右小括号表示 `part2.tex` 文件已被关闭。接下来在主文件 `sumfile.tex` 的第 10 页上又遇到一个错误, 因为这里并没有 `)` 表示该文件已被关闭。错误是在这个文件的第 217 行上遇到。

D.3 L^AT_EX 错误消息清单

下面列出 L^AT_EX 错误消息, 分成普通、宏包和字体错误消息三组。在每组中, 按错误指示的字母顺序排序。然后给出可能导致该错误的描述和解决方法。

有许多消息是新出现在 L^AT_EX 2_ε 中的, 也有一些相对于 L^AT_EX 2.09 版本稍有些改变。在 L^AT_EX 2_ε 中的所有消息都有一个前缀词 `! LaTeX Error:`, 而在 2.09 版本中错误指示只是前缀惊叹号, 并且位于第三行上。如果这是两者的唯一差别, 那么我们就显示 2.09 版本的内容。

当不同版本有其他不同时, 这里就进行显式说明。

D.3.1 L^AT_EX 普通错误消息

下述错误消息来源于那些没有调用字体选择或者定义, 或者不涉及 L^AT_EX 程序类和宏包文件 (附录 C) 的特殊功能时所出现的问题。

```
! LaTeX Error: ... undefined.
```

`\renewcommand` 或者 `\renewenvironment` 的参数值在前面还没有定义。应在前面插入相应的 `\new...` 命令。

```
! LaTeX Error: \< in mid line.
```

`tabbing` 环境中的 `\<` 命令出现在一行的中间。但这条命令只能位于一行的开头 (4.6.3 节)。

```
! LaTeX Error: Bad \line or \vector argument.
```

`\line` 或 `\vector` 命令的第一个参数值指定直线和箭头的倾角。这条消息说明所选择的角非法。见 6.4.3 和 6.4.4 节。

```
! Bad use of \\. (LATEX 2.09)
```

(这条消息在 L^AT_EX 2_ε 中已经被 `There's no line here to end` 取代。)

```
! LaTeX Error: Bad math environment delimiter.
```

L^AT_EX 遇到了一条 `\begin` 命令, 它不知道该环境名称。这可能是由于输入错误造成的。只要在处理时用反应 I 跟上正确的名称就可以校正这个错误。(这并没有改变源文件, 错误仍然还在那儿。)

```
! LaTeX Error: File '...' not found
Type X to quit or <RETURN> to proceed,
or enter new name. (Default extension: ...)
Enter file name:.
```

用输入命令上载一个文件, 但是却没有找到这个文件。这里可以输入一个新的文件名, 或者退出, 或者不管它继续处理下去。如果给出了正确的文件名, 而这个文件不存在, 那么可能是由于文件不在 L^AT_EX 寻找文件的路径中。确保把文件放在其中的一个正确目录中。此时用户有机会从键盘输入一个替代文件, 或者改正输入错误。输入文件名, 加上可省的扩展名, 然后按 <回车>。L^AT_EX 给出一个与要求相同的可省扩展名。文件名的基本部分并没有默认值。

```
! LaTeX Error: Float(s) lost.
```

`figure` 或 `table` 环境或者 `\marginpar` 命令在竖直盒子 (`\parbox` 命令或者 `minipage` 环境) 中给出, 或者这些结构出现在 L^AT_EX 命令 (例如脚注) 中, 而这条命令内部被竖直盒子调用。L^AT_EX 是在输出一页后才会发现这个错误, 因此实际的来源可能在指定有错文本前面好几行。这种错误的结果是有许多表格、插图或者边注丢失, 但并不是它们导致这个错误。

```
! LaTeX Error: Illegal character in array arg.
```

`tabular` 或 `array` 环境中包含一个未知的列格式字符 (6.6.1 节), 或者在 `\multicolumn` 命令中做为第二个参数值的格式项有错。

```
! LaTeX Error: \include cannot be nested.
```

在一个要用 `\include` 命令读入的文件中又用了命令 `\include`。所有的 `\include` 命令都必须在主文件中调用 (8.1.2 节)。

```
! LaTeX Error: LaTeX2e command ... in LaTeX 2.09 document.
```

如果正处在兼容模式 (即文档中用的是 `\documentstyle` 命令, 不是 `\documentclass` 命令), 却使用了 L^AT_EX 2_ε 命令, 就会出现这种错误。这些命令有 `\LaTeXe{}`, `\usepackage`, `\ensuremath`, `lrbox` 环境, 还有相应于 `\newcommand` 和 `\newenvironment` 命令增强的一些新语法。出错的原因是在兼容模式中作者可以确保文档适用于 L^AT_EX 2.09, 这样可以把它送给只有原来版本的用户手中。

```
! LaTeX Error: Lonely \item--perhaps a missing list environment.
```

在列表环境 (4.3 节) 外面用了 `\item` 命令。这要么是因为把 `\begin` 中的环境名拼写错了, 并且没有从键盘上改正它, 要么是忘记了 `\begin` 命令。

```
! LaTeX Error: Missing @-exp in array arg.
```

`tabular` 或 `array` 环境的列格式化参数值包含了 $\@$ 符号, 但后面没有必须的放在大括号 `{ }` 内的文本 (关于 $\@$ -表达式见 6.6.1 节), 或者在 `\multicolumn` 命令中的第二个参数里发生同样的错误。

! LaTeX Error: Missing begin{document}.

这要么是忘记输入 `\begin{document}` 命令, 要么是在文档的导言中有可打印的文本。对于后者, 这有可能是由于一个不合语法的声明造成的, 例如命令的参数值没有放在大括号 `{ }` 内, 或者命令名前面没有 `\`。

! LaTeX Error: Missing p-arg in array arg.

`tabular` 或 `array` 环境中的列格式化参数值包含 `p` 符号, 但没有必须与其一起出现的宽度定义 (6.6.1 节), 或者在 `\multicolumn` 命令的第二个参数值中发生同样的错误。

! LaTeX Error: No counter '...' defined.

! No such counter. (LaTeX2.09)

调用 `\setcounter` 或者 `\addtocounter` 命令, 但所引用的计数器并不存在。这很可能是名称被输错了。如果发生错误时正在读 `.aux` 文件, 而且计数器的名称肯定正确, 那么 `\newcounter` 命令就可能是在导言外面给出的。因此我们强烈建议总是把 `\newcounter` 命令放在导言中。(如果其他的 LaTeX 计数器命令用的是一个没有定义的计数器名称, 那么就会显示出一长串有趣的 TeX 错误消息。)

! No theorem environment '...' defined. (LaTeX2.09)

(在 LaTeX 2_ε 中, 这条消息被 `No counter defined.` 代替)

在 `\newtheorem` 命令中给出了可省参数值, 它指出另一个定理类型的声明, 以使两者共享同一个计数器, 但是这另一个定理结构并不存在 (4.5 节)。这要么是写错了定理的名称, 要么就是前面还没有定义。

! LaTeX Error: No \title given.

在给出 `\title` 声明之前就用了 `\maketitle` 命令。

! LaTeX Error: Not in outer par mode.

把 `figure` 或 `table` 环境或者 `\marginpar` 命令放在数学模式或者竖直盒子 (`\parbox` 命令或者 `minipage` 环境) 里面, 会导致这种错误。对于第一种情形, 通常是由于忘记了数学切换开关命令。

! LaTeX Error: Page height already too large.

用 `\enlargethispage` 命令来扩大页面的竖直尺寸, 但是 LaTeX 认为这个尺寸太大了。

! LaTeX Error: \pushtabs and \poptabs don't match.

在 `tabbing` 环境中 `\poptabs` 命令与前面已经出现的 `\pushtabs` 命令在数目上不相等 (4.6.4 节)。

! LaTeX Error: Something's wrong--perhaps a missing \item.

导致这个问题的原因很可能是由于列表环境 (`list`, `enumerate` 或者 `description`) 中的文本不是由 `\item` 命令开始的。如果在 `thebibliography` 环境中没有参数值 {样本标签}(4.3.6 节) 时也会出现这种错误。

! LaTeX Error: Suggested extra height (...) dangerously large.

当用 `\enlargethispage` 命令扩大页面的竖直尺寸时, 超出了 L^AT_EX 认为合理的范围。

! LaTeX Error: Tab overflow.

在 `tabbing` 环境中最后一个 `\=` 命令超过了在 L^AT_EX 中可接受的制表位数目。

! LaTeX Error: There's no line here to end.

! Bad use of \\. (L^AT_EX2.09)

在 `\par` 或者空行后面调用命令 `\newline` 或者 `\\`, 因为在这里它们没有任何意义。如果要在这一行插入额外竖直间距, 应该用 `\vspace` 命令。

! LaTeX Error: This may be a LaTeX bug.

这条消息说明 L^AT_EX 已完全糊涂了。这可能是由于前面出现错误时, 用户的反应是按了〈回车〉。对于这种情形, 应该用 `I\stop`, `X` 或者 `E` 把处理过程终止下来, 纠正前面的错误。虽然有可能 (但可能性不大) 这确实是 L^AT_EX 本身的一个漏洞。如果这是处理过程中的第一条错误, 而文本看起来却令人满意, 那么这个文件应该保存起来, 交给计算中心, 以供进一步研究。

! LaTeX Error: Too deeply nested.

列表环境 (`description`, `itemize`, `enumerate` 或者 `list`) 彼此嵌套的层次太多。可以这样嵌套的最多层数与安装版本有关, 但应至少为四层。

! LaTeX Error: Too many columns in eqnarray environment.

在 `eqnarray` 环境中每行只能有三列。出现这种错误, 可能是因为你忘记用 `\\` 开始新行, 或者在一行里多放了 `&`。

! LaTeX Error: Too many unprocessed floats.

在一页上有太多的 `\marginpar` 命令时, 会出现这个错误。然而, 更可能是因为 L^AT_EX 维持了超过其能力的太多的 `figure` 和 `table` 浮动对象。这种情况可能发生在浮动对象被输出之前, 提交了太多对象 (6.7 节)。对于这种情况, 就应该在文本中稍后加进最后这个对象。另外一种可能的原因是插图或者表格在正常的页面中放不下, 而它适合于放在文本结尾的特殊浮动页或者 `\clearpage` 或 `\cleardoublepage` 命令后面。由于插图和表格的输出顺序与输入顺序是相同的, 因此这样的一个对象就会阻塞整个队列。 `\clearpage` 或 `\cleardoublepage` 命令就可以化解这种阻塞。

! LaTeX Error: Undefined tab position.

在 `tabbing` 环境中用 `\>`, `\+`, `\-` 或者 `\<` 命令把制表位移动到了一个不存在的地方 (4.6 节)。

! LaTeX Error: \verb ended by end of line.

行内原文照排命令 `\verb+...+` 中的文本超过一行文本。在 $\text{\LaTeX}2.09$ 中这是可以的, 但在 $\text{\LaTeX}2_{\epsilon}$ 中, 这是禁止的, 因为这样很容易捕捉一个常见的错误: 少掉了结束字符。为了避免这种错误, 需要确保在初始字符和结束字符之间的文本都在一个输入行上。

! LaTeX Error: \verb illegal in command argument.

`\verb` 命令不能用在除 `\index` 和 `\glossary` 之外所有命令的参数值中。例如它就不能用在章节标题或者脚注中。

D.3.2 \LaTeX 宏包错误

那些用来控制类和宏包文本的特殊程序设计命令 (附录 C) 有它们自己的错误消息集。如果这些功能出现了个严重错误, 那么很少有用户能解决这个问题, 只有把它告诉文件的作者了。另一方面, 有些错误是源于对所提供的文件和选项不正确使用造成的。

通常类和宏包遇到其感到奇怪的文本和内容时, 会有它自己的错误或警告消息。在给出这些消息时, 会有类或宏包的名称。例如, 宏包 `mypack` 可能显示出如下错误:

```
Package mypack Error: cannot mix options 'good'
(mypack)                and 'bad'.
```

当按了 H 时, 应该会给出帮助消息。显然我们不可能在这里解释这些错误 (警告) 消息, 因为它们完全与所考虑的宏包有关。

! LaTeX Error: \LoadClass in package file.

在宏包文件中调用了 `\LoadClass` 命令, 这是不允许的。只能从一个类文件调用另一个类文件。

! LaTeX Error: Option clash for package

指定的宏包被重复调用, 而且两次所用的选项不同。宏包文件只能被调用一次, 第二次调用被忽略。因此, 如果 `\usepackage` 或 `\RequirePackage` 命令调用的是同一个宏包, 但是选项不同, 就会导致冲突。此时需要尽力安排一致的选项。按 H 可以得到帮助, 它会显示出这两组冲突选项。

! LaTeX Error: \RequirePackage or \LoadClass in Options Section.

这两条命令不能位于由 `\DeclareOption` 命令生成选项的类或宏包定义中。选项应该设置一些标志或者其他指示, 以供在调用这里所考虑的命令之前进行测试。

! LaTeX Error: This file needs format '...' but this is '...'

`\NeedsTeXFormat` 命令指定了一个与正在使用的格式不同的格式。所谓格式, 就是预先保存起来的指令集, 用来确定 \TeX 运行的类型。对于 $\text{\LaTeX}2_{\epsilon}$, 指定的格式名称必须是 `LaTeX2e`。错误所指的意思是用当前的格式根本不能处理这个文件。

! LaTeX Error: Two \documentclass or \documentstyle commands

一个文档中只能有一条 `\documentclass` 或者 `\documentstyle` 命令。如果在主文档中确实只看到一条这样的命令，那么检查一下其他上载的文件，确保其中没有与之冲突的第二条命令。

! LaTeX Error: Two \LoadClass commands.

在类文件中包含多于一条的 `\LoadClass` 命令，而这是不允许的。所用的类文件一定在编写上有问题。

! LaTeX Error: Unknown option '...' for package '...'.

在 `\usepackage` 命令中指定了一个选项，但是相应的宏包中并没有定义这个选项。查看一下宏包的指令，以及有没有输入错误。

! LaTeX Error: \usepackage before \documentclass.

`\usepackage` 命令不能出现在 `\documentclass` 的前面。必须在所有宏包之前上载类文件。

D.3.3 L^AT_EX 字体错误

下面的错误可能在应用新字体选择框架 (8.5 节) 定义或选择字体时出现。这些消息中有些会说字体还没有正确地建立，那么这可能是因为字体描述文件已崩溃或者有错。对于这两种情形，系统管理员必须找一组正确的文件，进行重新安装。

! LaTeX Error: ... allowed only in math mode.

类似于 `\mathbf` 这样的数学字母表命令用在了文本模式中。这可能是忘记了一个 `$`。

! LaTeX Error: Command '...' not defined as a math alphabet.

`\Set..` 或 `\Declare..` 命令以数学字母表名称做为参数，但在用这两类命令时，指定了一个不存在的字母表名称。数学字母表名称在使用之前，必须先用 `\DeclareMathVersion` 命令进行声明。

! LaTeX Error: Command ... not provided in base LaTeX2e.

有一些符号是基本 L^AT_EX 2.09 的一部分，但 T_EX 中并不存在，那么它们不再自动包含在 L^AT_EX 2_ε 中。这时需要利用软件包 `latexsym`。

! LaTeX Error: \DeclareTextComposite used on inappropriate
command

`\DeclareTextComposite` (8.5.9 节) 用来重定义特定编码中一个已存在的重音命令，以显示单个字符。如果这个重音命令并不存在，那就会给出这条错误消息。

! LaTeX Error: Encoding scheme '...' unknown.

进行了一个使用不存在的编码方案的字体声明或者选择命令。这很可能是由输入错误导致的。

! LaTeX Error: Font family '...+...' unknown.

对字体编码和族组合调用了 `\DeclareFontShape` 命令, 但是在这种组合之前并没有用 `\DeclareFontFamily` 命令进行声明 (8.5.8 节)。

```
! LaTeX Error: Font ... not found.
```

找不到具有指定属性的字体, 也无法进行适当的替换。这时会用由 `\DeclareErrorFont` 定义的字体。

```
! LaTeX Error: Math alphabet identifier ... is undefined
in math version '...'.
```

调用了一种对当前数学变体还没有定义的数学字母表 (8.5.5 节)。这就是说, 虽然已经用 `\DeclareMathAlphabet` 创建了字母表, 其形状属性为空, 但没有用 `\SetMathAlphabet` 声明把它定义成可以在指定数学变体中可用的数学字母表。

```
! LaTeX Error: Math version '...' is not defined.
```

`\Set..` 或 `\Declare..` 命令以数学变体名称做为参数, 但在应用这两类命令时, 指定了一个不存在的字母变体名称。数学变体名称在使用之前, 必须先用 `\DeclareMathVersion` 命令进行声明。

```
! LaTeX Error: *** NFSS release 1 command ... found
*** Recovery not possible. Use ....
```

用了第一版 NFSS 中的一条命令, 而这条命令现在不再可用了。用建议的替换命令。

```
! LaTeX Error: Not a command name: '...'.
```

`\DeclareMathAccent` 命令的第一个参数值必须是一个命令名称, 如 `\acute`。这时很可能是忘记输入反斜杠 `\` 了。

```
! LaTeX Error: Symbol font '...' not defined.
```

在 `\Set..` 或 `\Declare..` 命令中指定了一个不存在的符号字体名称。这两条命令需要符号字体名称做为参数值, 但是其中的字体名称在做为符号字体使用之前, 必须用 `\DeclareSymbolFont` 进行声明。

```
! LaTeX Error: The font size command \normalsize is not defined:
there is probably something wrong with class file.
```

在类文件中必须定义 `\normalsize` 命令, 这条命令定义文档中文本的标准尺寸。这是根据 L^AT_EX 2.09 中的经验进行的一个修改。如果类文件中没有进行这个定义, 而只是定义了 `\@normalsize`, 那就必须进行修正。如果没有 `\documentclass` 命令, 也会出现这条消息, 因为空的类文件当然是错误的。

```
! LaTeX Error: This NFSS system isn't set up properly.
```

在 `.fd` 文件中的字体描述有错误, 或者没有由 `\DeclareErrorFont` 声明的合法字体。这是一个严重的错误, 应向系统管理员报告这一信息。

```
! LaTeX Error: Too many math alphabets used in version ....
```

只可能有多达 16 个的数学字母表, 这是由 TeX 本身设定的限度。其他数学字母表定义都会被忽略。

! LaTeX Error: Unknown symbol font ‘...’.

用一个不存在的符号字体名称做为了 `\DeclareSymbolFontAlphabet` 命令的参数值。这里的字体名称必须在使用前用 `\DeclareSymbolFont` 进行声明。

D.4 TeX 错误消息

本节列出一些比较常见的 TeX 错误消息, 按错误指示进行字母排序, 而且同时给出一个导致该错误的简短描述。这里每条消息前面只是以惊叹号开始。由于这些消息是由 TeX 生成的, 因此在 L^ATeX 2.09 和 L^ATeX 2_ε 之间没有任何差别。

! Counter too large.

做为 TeX 错误, 这个消息是说, 当脚注标记用的是字母或符号时, 计数器的值超过了 26 或 9。如果在标题上有很多 `\thanks` 命令时也可能出现这个错误。

! Double subscript.

在数学公式中一个变量后面有两个下标, 例如, x_{2_3} 或 $x_{\{2\}_{\{3\}}}$ 。而要得到 x_{2_3} 的正确方法是 `x_{2_3}` 或者 `x_{\{2_{\{3\}}\}}`(5.2.2 节)。

! Double superscript.

在数学公式中一个变量后面有两个上标, 例如, x^{2^3} 或 $x^{\{2\}^{\{3\}}}$ 。而要得到 x^{2^3} 的正确方法是 `x^{2^3}` 或者 `x^{\{2^{\{3\}}\}}`(5.2.2 节)。

! Extra alignment tab has been changed to \cr.

在 `tabular` 或 `array` 环境中的一行里包含比列定义时还要多的 `&` 命令。这个错误可能是由于忘记在前一行尾部输入 `\\` 造成的。

! Extra }, or forgotten \$.

在数学模式中, 要么忘记输入左大括号 `{`, 要么就是多输入了右大括号 `}`。另外一种可能就是忘记输入数学模式切换命令, 如 `$`, `\[`, 或者 `\(`。

! Font ... not loaded: Not enough room left.

文本处理中要求上载的字符字体超过了由于内存限制 TeX 所能控制的数量。如果文档某一部分需要不同的字体, 那你可以把它分成几部分, 分别进行处理。

! I can't find file ‘...’.

这条消息实际上已经被 L^ATeX 2_ε 的等价消息 `File ... not found` 取代, 而且作用也非常相似。要读入指定的文件, 但是在搜索目录中却找不到这个文件。这时用户有机会输入另一个替换文件名称 (或者修改输入错误)。在显示出错误指示后, TeX 会问另一个要读入的文件名称:

Please type another input file name:

这就是等你再输入一个新名称,并按〈回车〉键。这与 \LaTeX 2_ϵ 消息有一点儿不同,默认的文件扩展名总是 `.tex`,并不要求输入扩展名。

`! Illegal parameter number in definition of`

这个错误很可能是由于不正确应用 `\newcommand`, `\newenvironment`, `\renewcommand`, `\renewenvironment` 命令造成的,即其中的替换字符 `#` 用法不对。这个字符在定义文件中只能以 `#n` 形式出现,这里的 `n` 就是介于 1 到在命令中指定的参数值个数之间的值。而要在文本中用到 `#` 字符,就必须用 `\#`。如果在后一个参数值 `{end.def}` 中用到了替换字符(7.4 节),也有可能導致这个错误。

`! Illegal unit of measure (pt inserted).`

如果这条消息恰好是出现在

`! Missing number, treated as zero.`

错误消息的后面,那么问题就是由于前一个错误造成的(见下面)。否则,错误就在于 \TeX 此时要求一个长度定义,但是只给出一个数,而没有长度单位。这通常是由于要把长度设为零,只给出了 0,而不是 `0pt` 或 `0mm`。如果是这种情形,那么用反应〈回车〉,就会得到正确的结果,因为对于零值,任何长度都得到同样的长度定义。如果完全忘记输入长度单位时,也可能出现这种错误。

`! Misplaced alignment tab character &.`

不能把单个字符命令 `&` 放在 `tabular` 或 `array` 环境外面的普通文本中。可能的动机是本来想显示 `&`,那么就应该输入为 `\&`。在响应错误消息时,可以用 `I\&` 来订正这一点。

`! Missing control sequence inserted.`

这个错误很可能是由于在 `\newcommand`, `\renewcommand`, `\newlength` 或 `\newsavebox` 命令的第一个参数值前面漏掉了反斜杠 `\`。只要用〈回车〉做为反应,就会正常地结束处理过程,因为 \TeX 假定你是少掉了反斜杠,从而替你补上了。

`! Missing number, treated as zero.`

这个错误很可能来源于需要一个数值或长度做为参数值的命令,而却少掉了这个参数值。或者也有可能是一条命令有可省参数值,而其后所接的文本开头就是 `[`,也可能造成这种错误。最后一种可能是在长度或者 `\value` 命令前面有 `\protect` 命令。

`! Missing { inserted.`

`! Missing } inserted.`

当显示这两条消息之一的时候,就表明 \TeX 已完全糊涂了。这时显示出来的行号往往并不是出现错误的地方,这是由少掉一个左大括号或右大括号造成的。如果错误不是很明显,那就按〈回车〉让处理继续下去,从打印出来的结果中推断错误出现的地方。

`! Missing $ inserted.`

很可能是在普通文本中用了一个只能出现在数学模式中的符号或命令而造成这种错误。查看一下在第五章中列出的那些除非有特别说明,否则只能用在数学模式中的命令。

如果在数学公式中插入了一个 `\mbox` 命令, 那么其参数值就暂时从数学模式切换到文本模式。如果在数学公式中出现一个表示新段的空行, 或者在公式的尾部没有必要的 `$` 符号也会出现这种错误。

! Not a letter.

`\hyphenation` 命令的单词列表中包含一个不被认为是字母的字符, 例如 `\'e` 这样的重音命令。这样的单词可以通过显式插入可选连字符来给出建议断点 (3.6.1 和 3.6.2 节)。

! Paragraph ended before ... was complete.

命令的参数值中包含空行或者 `\par` 命令, 而这是不允许的。很可能是漏掉了右大括号 `}`。

! \scriptfont ... is undefined (character ...).

! \scriptscriptfont ... is undefined (character ...).

! \textfont ... is undefined (character ...).

在 $\text{\LaTeX}2.09$ 中, 当数学公式使用了一个符号, 而其字体并不是针对于数学模式而设计的, 如 `\sf(sans serif)`, 就会显示这种消息。这样的字符字体必须用 `\load{尺寸}{样式}` 命令才可以使用, 如 `\load{\normalsize}{\sf}`。如果这样的公式出现在脚注中, 就必须事先给出 `\load{\footnotesize}{\sf}` 命令。在 $\text{\LaTeX}2_{\epsilon}$ 中, 只要用的是 `\mathsf` 这样的数学字母表命令, 就不会再出现这种问题。

! TeX capacity exceeded, sorry [...].

TeX 在进行文本处理时, 会在计算机内存中建立起各种存贮缓冲区。当其中某一个缓冲区已用完, 从而不能再添加信息时, 就会显示这条消息。缓冲区的名称及其最大尺寸显示在错误指示的中括号内。显示完这条消息, TeX 的处理过程被终止。无论处理的文本有多复杂, 这个问题的来源也多半不会是由于内存不足造成的, 更可能是文本自身的错误。在 D.6 节描述的方法可以用来探察真正的错误原因。

下面描述一下各种缓冲区, 来帮助你确定是否真的是因为 TeX 分配缓冲区能力太小, 以及应该怎样校正这个错误。

`buffer size` 可能是由于章节命令、`\caption`、`\addcontentsline` 或 `\addtocontents` 命令中的参数值太长造成的。这条消息通常只有当 `\end{document}` 命令被处理时才会出现, 但是在遇到 `\tableofcontents`、`\listoffigures` 或 `\listoftables` 时也有可能出现。避免这种错误的方法就是对那些太长的标题文本, 不妨使用相应命令的可省参数值, 给出短标题 (3.3.3 和 6.7.6 节)。因为这样的太长的项出现在目录表也是很烦人的, 总是要把它缩短。一旦在源文本中进行了这种修正, 必须删除已有的 \LaTeX .aux 文件。

在 PC 机上如果用的是字处理程序, 而不是文本编辑器来生成源文本, 有可能出现这个问题。因为有的字处理程序在编辑一个段落文本时, 即使在屏幕上分成几行文本显示的, 但在文件中却是整个段落放在单独一行上。

exception dictionary 用 `\hyphenation` 命令输入的连接符例外清单太大了。那些使用频率很小的单词应该去掉，而用 `-` 进行显式地指定单词断点。

hash size 在源文件中包含了太多的命令定义或者交叉引用标志。这并不是说源文件需要所有这些命令，而可能是由于用户自己开发出一个很大的专用命令集，并把它存贮在一个文件中。在编辑 \LaTeX 文档时，不管用到用不到这些命令，都全部读入到当前文档中。

input stack size 这个缓冲区的溢出可能是命令定义中出现了错误。例如，如果用下面这样的命令定义：

```
\newcommand{\com}{One more \com}
```

就会得到 `One more {One more {...One more \com}...}`，永远这样持续下去，因为它自己调用了自己。实际上它并不会永远持续下去，直至缓冲区变满。

main memory size 这个缓冲区包含当前正在处理页面中的文本。如果调用了递归定义命令，它也会溢出。然而，更常见的原因是：

- (1) 在一页上定义了很多相当复杂的命令；
- (2) 在一页上有太多的 `\index` 或 `\glossary` 命令；
- (3) 页面自身太复杂，以致于充满分配的缓冲区空间。

对于前两种情形的解决方法非常简单：减少该页的命令定义或 `\index` 与 `\glossary` 命令。对于第三种情形，可能是由于有太长的 `tabbing`, `tabular`, `array` 或 `picture` 环境，或者等待输出的浮动对象（插图或表格）发生了堵塞。

为了验证内存溢出是否确实是由于复杂的页面造成的，那么在出现溢出的地方前面加上一条 `\clearpage` 命令。如果在接下来的处理运行中不再出现这种错误，那么就可以确定，这真的是由于该页面对 \TeX 而言确实太复杂了。然而，如果仍然有溢出，那么就可能是由源文本中的错误造成的。必要的话，可以用 D.6 节给出的方法确定错误的位置。

如果确实是由于，对 \TeX 处理面言，页面太复杂了，那就必须想法简化。然而要注意到在输出页面之前要把最近的那整个段落处理完，即使分页点可能出现在这个段落的开始部分。利用 `\newpage` 命令，可能会解决这个问题，因此应该在进行烦人的文本重组织之前，尝试一下这条命令。如果错误是由于浮动对象阻塞造成的，那么把浮动对象移到后面文本中，或者改变一下定位参数值 (6.7.3 节)，可能会解决这个问题。如果整个文本还没有结束，那么现在可以尝试用 `\clearpage`，以清除被阻塞的浮动对象，然后在完成文本后给出浮动对象的一个更合理的顺序。

pool size 很可能是由于有太多的命令定义和标签，或者是其名称太长。缩短一下名称长度看看。在诸如 `\setcounter`、`\newenvironment` 或 `\newtheorem` 等命令中的计数器命令参数值漏掉了右大括号 `}` 时也会出现这种错误。

save size 当命令、环境和声明的范围嵌套层数太多时，这个缓冲区就会溢出。例如，在命令 `\multitup` 的参数值中包含 `picture` 环境，而这个环境中又在另一个 `\multitup` 命令中用了 `\footnotesize` 声明，以此类推。这样的嵌套必须简化，除非实际导致问题的原因是因为忘记了右大括号 `}`，而使得结构变得相当复杂。

关于在 emTeX 中如何改变上述内存布局值的操作方法，见 9.3.5 节。

`! Text line contains an invalid character.`

在源文本中包含一个特殊符号， TeX 无法识别。这可能是编辑器本身的问题，是它插入了这个多余的字符。如果检查源文本时无法看到这个字符，那就只有寻求计算中心的帮助了。

`! Undefined control sequence.`

每个 TeX 用户都会经常遇到这条错误消息。它通常是由不正确输入命令名称造成的。可以在处理过程中用 I 及正确的命令名称，再加上 \langle 回车 \rangle 进行更正。如果正确地输入了一个 $\text{L}\text{a}\text{T}\text{E}\text{X}$ 命令，那就可能是由于在不正确的环境中调用造成的，因为在这个环境中这条命令是不允许使用的（即还没有定义）。

`! Use of ... doesn't match its definition.`

如果 \cdots 是一个 $\text{L}\text{a}\text{T}\text{E}\text{X}$ 命令的名称，那它就很可能是 6.3 和 6.4 节中的图形环境中的命令调用时参数值语法不对。如果名称是 $\backslash @array$ ，那么在 tabular 或 array 环境中有一个有错的 $@$ -表达式 (6.6.1 节)。这可能是由于在 $@$ -表达式中有一个脆弱命令，但没有前缀 $\backslash protect$ 命令。

`! You can't use 'macro parameter #' in ... mode.`

在普通文本中用了特殊符号 $\#$ 。这可能是需要用 $\backslash \#$ 生成 $\#$ 自身时造成的。可以在处理过程用 $\text{I}\backslash \#$ 加 \langle 回车 \rangle 纠正这个错误。

D.5 警告

TeX 和 $\text{L}\text{a}\text{T}\text{E}\text{X}$ 错误都会使得处理过程暂时停下来，并等待用户的反应，或者程序完全中止。而另一方面，警告只是提示用户输出结果中可能有些不对头，需要进一步修正。警告消息连同其所在的页码一起显示在屏幕上，但不会使处理过程停下来。当然这些消息也会同时写到 .log 文件中，这样处理结束后可以查看该文件，甚至打印出来。 TeX 或者 $\text{L}\text{a}\text{T}\text{E}\text{X}$ 都有可能导致警告。

D.5.1 普通 $\text{L}\text{a}\text{T}\text{E}\text{X}$ 警告

$\text{L}\text{a}\text{T}\text{E}\text{X}$ 的警告是以 'LaTeX Warning:' 为标志的，它位于消息的开头，后接警告消息。

`LaTeX Warning: Citation '...' on page ... undefined on
input line`

在 $\backslash cite$ 命令中的关键词并没有对应的 $\backslash bibitem$ 命令进行定义 (4.3.6 和 8.3.2 节)。

`LaTeX Warning: Citation '...' undefined on input line`

在 $\backslash nocite$ 命令中的关键词还没有定义。

`LaTeX Warning: Command ... has changed.`

`Check if current package is valid.`

用 `\CheckCommand` 语句来测试一个命令是否已经具有特定的定义。如果这个测试失败, 就会给出这条警告消息。这可以用来确认给定宏包的设计者到底想做什么。

LaTeX Warning: Float too large for page by ..pt on input line ...

浮动对象 (figure 或 table 环境) 太大, 在一页上放不下来。当然浮动对象总是会被显示出来的, 但是会超出正常页面的边界。

LaTeX Warning: 'h' float specifier changed to 'ht'.

浮动对象的定位指定 'h' 表示 '这里' (6.7.3 节), 并不一定能精确地放在浮动对象所处的位置上。只有在当前页面有足够的空间时, 才会把它放在 '这里', 否则它会出现下一可用页面的顶部。这个消息就是对这一事实的提醒。

LaTeX Warning: inputting '...' instead of obsolete '...'.

有些专门为 L^AT_EX 2.09 编写的宏包显示要求输入特定的文件, 例如 `article.sty` 已经有一个新的等价文件, 但名称不同 (此时为 `article.cls`)。那么原文件名称就对应一个空文件, 这里给出这个消息, 要求输入正确的文件。

LaTeX Warning: Label '...' multiply defined.

两条 `\label` 或 `\bibitem` 命令定义的标记有相同的名称 (8.3.1 和 8.3.2 节)。即使进行了改正, 这条消息还会再显示一次, 因为这次能从上次运行生成的 `.aux` 文件中找到重名标记。当第三次运行时, 它应该没有了。

LaTeX Warning: Label(s) may have changed.

Return to get cross-references right.

从 `\ref`, `\pageref` 和 `\cite` 命令中得到的输出可能不正确, 因为在处理中它们的值已发生了变化。这就必须再运行一次 L^AT_EX, 以使用正确的值。

LaTeX Warning: Marginpar on page ... moved.

在指定页上的边注已经被向下移动了, 以防边注之间离得太近。这就是说这个边注并不是从 `\marginpar` 命令实际所位于的那行开始显示的。

LaTeX Warning: No \author given.

调用了 `\maketitle` 命令, 但是前面却没有 `\author` 命令。与没有 `\title` 命令不同的是, 这并不是一个错误, 只是它觉得有点儿奇怪。之所以显示这个消息, 就是以防你忘记输入作者信息。

LaTeX Warning: Optional argument of \twocolumn too tall on page..

`\twocolumn` 命令 (3.2.6 节) 开始新页, 并切换进入两列格式。在可省参数值中的文本以单列方式显示在两列文本的顶部。如果这部分文本太长, 在一页中放不下来, 就会显示出这条警告。

LaTeX Warning: Oval too small on input line

在 `\oval` 命令中指定的尺寸太小, L^AT_EX 无法找到一个相应的四分之一圆周。

LaTeX Warning: Reference ‘...’ on page ... undefined on
input line

在 `\ref` 或 `\pageref` 命令中的标志名在前一次运行时还没有用 `\label` 命令进行定义 (8.3.1 节)。如果在下一次运行中还会出现这条消息，那就是少掉了相应的 `\label` 命令。

LaTeX Warning: Text page ... contains only floats.

这条消息指出，在指定页上，浮动对象样式参数已经把所有其他正常文本排除在外了。这并不是不好，但通常你需要自己看看这一页面的实际效果。

LaTeX Warning: There were multiply-defined labels.

如果有 `\label` 或 `\bibitem` 命令多次使用同名的标志，就会在 \LaTeX 运行结束时显示这条消息。在运行到每个重复标志附近时，也会显示一条警告消息。

LaTeX Warning: There were undefined references.

如果在前一次运行中，有的 `\pageref` 命令中的标志还没有定义，就会在 \LaTeX 运行结束时显示这条消息。在每次用到一个还没有定义的标志时也会显示一条警告消息。

D.5.2 \LaTeX 宏包警告

类和宏包警告主要就是当要求上载的文件版本和名称与实际不符，或者选项使用不当，或者不知道 `filecontents` 环境已经向一个文件中写入了某些文本时显示的消息。这里所提及的命令在 C.2.9 节有介绍。

类和宏包也可以有自己独特的警告消息。这里不可能列出这些消息，因为它们是与类和宏包密切相关的。

LaTeX Warning: File ‘...’ already exists on the system.
Not generating it from this source.

由于 `filecontents` 环境发现在系统中已存在一个同名的文件，因此它并没有从主文件中提取文本。

LaTeX Warning: Unused global option(s):....

在 `\documentclass` 声明中指定的选项是全局性的，即它要应用于所有的类和后面调用的宏包。然而，如果对某一个选项，所有的类或宏包都不能识别，就会显示出这条警告消息，然后列出没有用的选项清单。

LaTeX Warning: Writing file ‘...’

`filecontents` 环境 (C.2.9 节) 正从主文档中提取信息，并写到一个给定名称的文件中。

LaTeX Warning: You have requested class/package ‘...’,
but the class/package provides ‘...’.

由内部识别命令 `\ProvidesClass` 或 `\ProvidesPackage` 给出的类或宏包名称并不与 `\usepackage` 或 `\RequirePackage` 命令中的名称一致。

LaTeX Warning: You have requested, on input line ..., version
 '...' of class/package ...,
 but only version '...' is available.

由内部识别命令 `\ProvidesClass` 或 `\ProvidesPackage` 给定的类或宏包日期, 早于 `\usepackage` 或 `\RequirePackage` 命令要求的日期。此时类或宏包可能不具备输入文件所要求的全部功能。

LaTeX Warning: You have requested releases '...' of LaTeX,
 but only release '...' is available.

你拥有的 \LaTeX 版本比某些输入文件中 `\NeedsTeXFormat` 命令要求的早 (C.2.1 节), 因此它可能不会提供该文件要求的所有功能。

D.5.3 \LaTeX 字体警告

字体警告是在调用 NFSS 命令 (8.5 节) 时出现的。其标志为 LaTeX Font Warning: 后接警告文本。

LaTeX Font Warning: Command ... invalid in math mode.

只能用在文本模式中的命令, 被用在数学模式中, 这条命令会被忽略。`\mathversion`, `\boldmath`, `\unboldmath` 和 `\em` 都会导致这条消息。也有一些命令会产生同样内容的错误消息。

LaTeX Font Warning: Command `\tracingfonts` not provided.

(Font) Use the 'tracefnt' package.

(Font) Command found: on input line

只有上载了 `tracefnt` 宏包 (205 页) 时才可以使使用字体跟踪诊断工具 `\tracingfonts`, 否则就会忽略该命令。

LaTeX Font Warning: Encoding '...' has changed to '...' for

(Font) symbol font '...' in the math version '...'.

要想在给定的数学变体中应用某一特定的符号字体, 需要暂时修改字体编码。

LaTeX Font Warning: Font shape '...' in size <...> not available

(Font) size <...> substituted.

没有为要求的尺寸和形状定义字体, 因此用了一个替换尺寸。

LaTeX Font Warning: Font shape '...' undefined

(Font) using '...' instead.

要求的形状属性是未知的, 或者还没有定义, 因此使用一个替换形状。

LaTeX Font Warning: *** NFSS release 1 command ... found

(Font) *** Update by using release 2 command

(Font) *** Recovery is probably possible.

使用了一条来自于 NFSS 第一版本的命令，而这条命令现在是不可用的。但是 \LaTeX 会尝试插入新的等价命令，并继续处理下去。

D.5.4 \TeX 警告消息

为了区分开 \TeX 警告消息与错误消息，在警告消息文本的前面没有任何前缀，处理过程也不会停下来。最常见的 \TeX 警告消息是：

```
Overfull \hbox ....
```

\TeX 无法以一种合理的方式断开该行，从而该行有部分内容伸展进右边界。消息中的其他内容可以给你一些帮助。

例如，下面是一条完整的警告消息：

```
Overfull \hbox (17.2122pt too wide) in paragraph at lines 4--6
[]\OT1/cmr/m/n/10 If T[]X can-not find an ap-pro-pri-ate
spot to di-vide a word at the end of the line, as right
here aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

从这条消息我们可以知道当前行长度约超过了 17.2 pt (6 mm)，这也就是进入右页边界的长度。这一行位于从 4-6 行所在的段落中。所用字体的属性标志为 `\OT1/cmr/m/n/10`。有问题行的文本是 `If right here aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa`。供选择的断词点用连字符表示，如 `di-vide`。最后那个单词无法断开，也就是它导致当前的问题。

解决这个问题的一种方法是利用建议断词点，即在单词内部插入 `\-`，例如我们这里可以用 `aaaaaaaa\ -aaaaaaaaaaaaaaaaaaaaa` 代替原来的输入。同样，在有问题单词前面插入 `\linebreak` 命令，或者把整个段落放在 `sloppypar` 环境中，也会去掉这条警告消息。

如果一行中的断词不是很好，但却只稍稍长一点儿，比如说是 1 pt 或更少，在绝大多数情形中很少会注意到它，从而我们就不必在意这一点。

```
Overfull \vbox ....
```

很少会出现这条消息。它表示 \TeX 不能很好的分页，文本伸展到了页面底部。 \TeX 通常宁可在一页上少放些内容，也不会放太多的东西。因此只有当一页上有一个太大 (超过 `\textheight` 的当前值) 的竖直盒子，例如一个长表格时，才会显示出这条警告消息。

```
Underfull \hbox ....
```

这是与 `Overfull \hbox` 警告相对的一条消息。当 \TeX 对一行进行了填充，以使得左右对齐后，却使得该行的单词间距太大，从而它认为结果不太令人满意时就会显示这条消息。这通常是 `sloppypar` 环境，`\sloppy` 声明或者 `\linebreak` 命令的后果。也有可能是对 `\\` 或 `\newline` 的不恰当应用造成的，例如连续给出了两条 `\\` 命令。警告中的其它消息由格式化结果不好的文本以及对单词间距糟糕程度的估计组成。

如果对于上面所考虑的 `Overfull \hbox` 例子，在 ... 处插入 `\linebreak` 命令，即 `as right\linebreak here`，那么警告消息为

```
Underfull \hbox (badness 5504) in paragraph at lines 4--6
[]\OT1/cmr/m/n/10 If T[]X can-not find an ap-pro-pri-ate
```

spot to di-vide a word at the end of the line, as right

这条消息说明, 在第 4 行到第 6 行上的段落包含了一个输出行, 其单词间距太宽, 不能令人满意。这行文本的内容为 `If, as right`, 当前字体为 `\OT1/cmr/m/n/10`。估计值 `badness 5504` 是对间距糟糕程度的度量, 这个值越小, 结果就越好。

为了对这个 `badness` 有一个大致的印象, 就必须知道它到底是如何计算出来的。所有的单词间距都有一个基本尺寸, 以及它可以伸展或收缩的最佳限度。对于每个输出行, \TeX 会求出达到理想伸展和收缩后的实际单词间距。 `badness` 的值为

$\text{badness} = 100 \times (\text{实际的收缩量} / \text{最佳的收缩量})^3$ 或者

$\text{badness} = 100 \times (\text{实际的伸展量} / \text{最佳的伸展量})^3$

这里 ‘实际的收缩量’ 和 ‘实际的伸展量’ 就是在每行两端加上或减去的距离。

通常 \TeX 容忍把单词间距伸展到 `badness=200`, 即真正的伸展是最佳伸展的 1.26 倍。在 `sloppypar` 环境或者 `\sloppy` 声明后, 文本行是可以无限伸展的。当 `badness` 超过 1000 时就会显示出 `Underfull \hbox` 警告消息, 这意味着伸展值为最佳值的 2.15 倍。如果上述公式结果超过 10000, 就简单地取 `badness` 为 10000。

在实际应用过程中, 可以容忍 `badness < 2000`, 即使这时不理想的单词间距是很容易注意到的。应该打印出结果看看到底怎样。

`Underfull \vbox`

已进行了分页, 并对顶部与底部进行了调整, 但 \TeX 认为段落间距可能不太令人满意。这里的 `badness` 数相应于 `Underfull \hbox` 警告消息中同名数量。

D.6 搜索顽固错误

有的时候, 你会遇到一个错误, 怎么也找不到出错的地方。对于这种可恶的错误, 我们推荐如下的搜索策略:

1. 把正在处理的文档复制两份, 一份做为旧版本, 一份做为工作版本 (原来的版本仍保留住, 在下面的搜索过程中不要改动它)。
2. 在工作版本文档中, 找到出错地方的最外层环境, 去掉一个或多个内层环境。如果其没有内层环境, 就缩短余下的文本。利用 \LaTeX 再次处理这个文件。
3. 如果仍然有那个错误, 就把缩短后的工作版本复制到旧版本中, 重复第 2 步。如果第 2 步中外层环境是 `\begin{document} ... \end{document}`, 那么就可以通过在某些点简单地插入 `\end{document}` 来缩短文本。也可以利用 `\iffalse... \fi` 把一块文本注释掉, 见 4.9 节。
4. 如果在缩短后的工作版本中不再有那个错误, 那就把旧版本内容复制到工作版本中, 这样错误仍然在这个版本中。这次比上次少去掉一些文本, 重复 2 到 4 步。
5. 如果按这种方法, 发现错误是在下一个最外层环境中, 那就重复从 2 到 4 的同样操作。

利用这种方法, 就可以把错误局限在一条命令或者只剩下很少一部分结构的环境中。如果尽管错误已被准确定位, 但仍然不知道为什么错了, 那就只好请求一些有经验人士

的帮助了。然而，通常只要一找到出错的地方，就很容易知道为什么错了。

有时也确实会即使已改正了错误，在下次运行 \LaTeX 时还会出现这条错误消息。这主要是因为 \LaTeX 通过辅助文件进行信息传递，而这些文件的更新是在当前处理过程结束。例如，若某一个章节命令中有错误，那么即使已修正了这个错误，在 `.toc` 文件中仍旧是那个有错误的条目。如果文档中包含 `\tableofcontents` 命令，那么 \LaTeX 就会在下次运行时读入这个 `.toc` 文件，从而仍然给出那条错误消息。只有当成功处理完一次文件后才会得到新的 `.toc` 文件。

对于这种情况，应当编辑 `.toc` 文件，去掉那个错误。如果没法这样做，那就删掉这个 `.toc` 文件，并用 \LaTeX 编译修正的文档两遍。如果错误是在 `\caption`、`\addcontentsline` 或者 `\addtocontents` 命令中，也要对 `.lof` 或 `.lot` 文件做同样的处理。

有时我们也必须删掉 `.aux` 文件，以杜绝虽然已改正了 `.tex` 文件中的错误，但仍然显示同样的错误。这时如果导言中有命令 `\nofiles`，用户必须更加小心，因为此时再运行一次 \LaTeX 也不会生成正确的 `.aux` 文件。

附录 E 计算机现代字体

当 Donald E. Knuth 发明 $\text{T}_{\text{E}}\text{X}$ 程序时, 也同时给它配备了丰富的字符集 (或称为字体), Knuth 称这些字体为 计算机现代字体(computer modern fonts, 简称为 CM 字体), 利用这些字体, 就不必依赖于所用计算机上的字体。在当时, 打印机字体质量并不是很高, 当然也不会一致。而利用他所提供的字体, $\text{T}_{\text{E}}\text{X}$ 就能在所有打印机上生成相同的高质量输出。

$\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 当然继承了这些字体, 因此这些字体已经成为由 $\text{T}_{\text{E}}\text{X}$ 或 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 生成的文档的标志。不过话又说回来, 并不是一定要这样, 因为 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 并不需要与任何特定的字体集合捆绑在一起, 特别是在当今新字体选择框架出现 (8.5 节) 之后, 新框架极大地简化了字体安装的步骤。

但是对绝大多数 $\text{T}_{\text{E}}\text{X}/\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 用户而言, 计算机现代字体仍然起着重要作用, 因此有必要对其做一详细介绍。即使我们在 NFSS 中只是指定了一种字体属性, 最终这些属性组合还是要与已有的某种字体名称联系在一起, 见 8.5.8 节。对于标准安装版本, 这些名称就来自本附录余下部分描述的字体集合。

E.1 简介

排版学就是对字样进行研究与分类的科学, 而字样指的是可以上溯到五个半世纪前由 Gutenberg 所发明的活字 (movable type)(不是指中国人发明的活字印刷)。从那时开始, 人们创建了相当多的字体族, 并把其分类为 Baskerville, Caramond, Univers, 等等。如此字体族中每个成员拥有同样的整体设计, 或者称基本式样, 但是可以有 slanted, 斜体、黑体或细体等变体; 当然, 它们可以有不同的尺寸。

对字体族的分类通常是根据某种规则进行, 主要的考虑原则是它们的用途。

Serif 字体 是一种在边缘有小水平线 (也称为衬线, serifs) 的字体, 这样眼睛看起来感觉很好。经验表明这种字体适于阅读, 因此通常用作正文字体。按 NFSS 的术语, 这些字体称为 罗马(Roman) 字体。

Sans serif 字体 一种缺少衬线的字体。具有赤裸外形的这种字体通常用在标题或者书眉中。比较 sans serif 和 regular text 这两种字体就可以知道它们的区别了。

固定字体 是一种具有相同宽度的字体, 它是从打字机字体演化而来, 进而进入到计算机字体的行列中。按照正统的观点来看, 在书籍印刷中是没有这种字体的用武之地的, 因为这时成比例字体 (字母 i 要比 m 窄) 占主导地位。

装饰性字体 是一种由于某些不寻常之处而显得突出的字体。通常用它来吸引眼睛的注意力。这一族在形状和宽度方面还不完整, 通常用在广告中。

数学字体 是数学作品中所需要的特殊符号全体。对它的进一步分类是无法与对文本字体的分类一起讨论的。

书籍设计人员必须决定要使用的字样族类型。我们有可能在正文用罗马 (有衬线的) 字体, 书眉用 sans serif 字体, 而且如果书中还有计算机程序代码, 就用一种固定字体表示这部分内容。最后, 如果作品中有数学内容, 还需要符号字体。

在计算机现代字体集合中包含所有上述种类的族。然而, 由于它们是在 NFSS 属性分类系统建立之前出现的, 所以 CM 字体命名法并不与这种框架完全一致。NFSS 系统把用户从必须记住 CM 字体名称的细节中解脱出来, 而只需要指定属性。当然, NSF 安装中必须有字体描述文件 .fd, 这个文件把属性组合转化为真实的字体名称, 或者进行某种过得去的替代。

E.2 T_EX 基本字体的分类

任何特定的字体都可以用在 L^AT_EX 中, 方法之一就是利用 `\newfont`(4.1.6 节) 命令直接给出字体的名称, 另外一种方法则是用 `\usefont`(8.5.1 节) 或 `\DeclareFixedFont`(8.5.4 节) 给出其属性。

所有 T_EX 字体文件的名称都是以 cm(表示 “Computer Modern”) 字母开头的, 后接一至四个字母描述字体的样式, 最后是一两个数字指定设计尺寸的点数。这也就是用在 `\newfont` 命令中的字体基本名。因此基本名的完整语法为

`cmxxnn` `xx`= 样式, `nn`= 设计尺寸

基本字体文件的扩展名为 .tfm, 含义为 “T_EX font metric”。

在 .tfm 文件中并没有包含生成符号的信息, 它只是由字符尺寸信息组成, 这些尺寸包括宽度、高度和深度。对于 slanted 字体, 每个字母还有 “倾斜校正”(3.5.1 节)。而且, 要为一些字母组合指定与通常不同的间距, 例如 AV 与 $\mathcal{A}\mathcal{V}$ 的区别, 或者有可能进行连写。最后, .tfm 中还包含字符的斜率 (对于非 slanted 字体, 这个值为零), 标准单词间距及其伸展和收缩量 (D.5.4 节), em 和 quad 间距的大小, 句子结尾处间距。数学和符号字体在其 .tfm 文件中还包含更多的信息。

`\newfont` 命令的一般语法是

`\newfont{\字体}{基本名scaled 因子}`

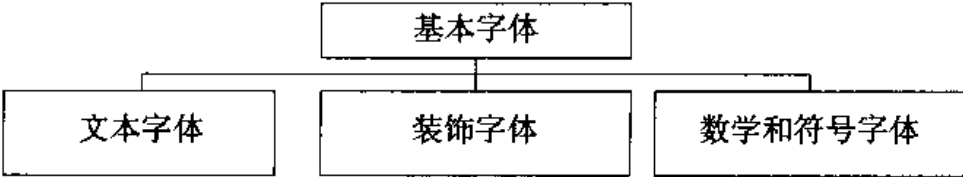
其中 基本名 指的是字体文件的外部基本名, 因子 就是对字体进行放大时所采用的放缩因子的 1000 倍。这样所得的新命令 `\字体` 就是一个声明, 它激活指定的字体, 作用范围为当前环境结束, 或者又调用了新的字体命令。如果所用的是 L^AT_EX, 那么 因子 可以取任何值, 因为处理时就用 .tfm 文件中的数乘上这个因子。

也可以用下述命令得到同样的效果:

`\DeclareFixedFont{\字体}{编码}{族}{序列}{尺寸}`

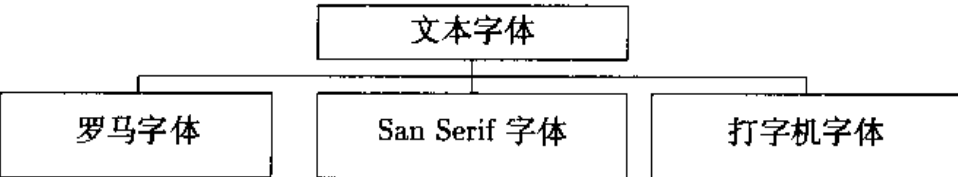
这条命令自动地把 `\字体` 定义成适当的字体, 必要时进行放缩。

下面我们给出 CM 字体的名称, 并说明它们如何遵从 NFSS 属性模式。它们的组织如下:



E.3 文本字体

用来设置普通文本的字体可以分为三族，其编码都为 OT1。



E.3.1 罗马字体族

罗马字体族就是有衬线的字体，间距成比例。这些字体通常充满了正文的主体。

cmr 族, OT1 编码						
形状 =	n	sc	sl	it	u	
序列						
m	cmr5	cmcsc10	cms8 cms9 cms10 cms12	cmti7 cmti8 cmti9 cmti10 cmti12	cmu10	
	cmr6					
	cmr7					
	cmr8					
	cmr9		cmbxsl10	cmbxti10		
	cmr10					
	cmr12					
	cmr17					
b	cmb10					
bx	cmbx5					
	cmbx6					
	cmbx7					
	cmbx8					
	cmbx9					
	cmbx10 cmbx12					

直立罗马字体

直立罗马字体组由在书籍印刷中最经常使用的标准字符集组成。字体样式标志为字母 r, 因此在这一族中所有文件名都以 cmr 开头, 后接以点为单位的设计尺寸。设计尺寸的变化范围从 8 到 17 pt. cmcsc10 字体也属于该族, 但形状为 sc. 这里的 csc 代表大写和小大写 (*capital and small caps*). 因为在该族中的大写字母与 cmr10 中的一样, 但小写字母是一种介于 cmr7 和 cmr8 之间尺寸的一种较小的大写字母。

cmr10

ABCDEFGHIJKLMNOPQRSTUVWXYZ ÆŒ Ø @#\$\$%&

abcdefghijklmnopqrstuvwxyz ff fi fl ffi ffi æœøßij 0123456789
 ΓΔΘΛΞΠΣΥΦΨΩ`´˘˙˚˛˜˝˞˟ˠˡˢˣˤ˥˦˧˨˩˪˫ˬ˭ˮ˯˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿?ii'""- - —*+/() []

cmcsc10

ABCDEFGHIJKLMNOPQRSTUVWXYZ ÆŒ Ø @# \$ % &
 ABCDEFGHIJKLMNOPQRSTUVWXYZ ↑ ↓ ' ; ; ÆŒ Ø Ñ 0123456789
 ΓΔΘΛΞΠΣΥΦΨΩ`´˘˙˚˛˜˝˞˟ˠˡˢˣˤ˥˦˧˨˩˪˫ˬ˭ˮ˯˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿>?<'""- - —*+/() []

仔细查看 cmcsc10 字体，并与 cmr10 进行对比，我们会注意到所有小写字母的高度是一样的（介于 cmr7 和 cmr8 的相应字母大小之间），但宽度要大一些，而且字母间距也要大些。另外连写被其他符号代替了。

cmu10

ABCDEFGHIJKLMNOPQRSTUVWXYZ ÆŒ Ø @# £ % €
 abcdefghijklmnopqrstuvwxyz ff fi fl ffi ffi æœøßij 0123456789
 ΓΔΘΛΞΠΣΥΦΨΩ`´˘˙˚˛˜˝˞˟ˠˡˢˣˤ˥˦˧˨˩˪˫ˬ˭ˮ˯˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿?ii'""- - —*+/() []

cmu10 字体实际上是一种没有倾角的斜体。这样就得到了一种直立的斜体字母。在 NFSS 框架中它占据了一种特殊的形状属性 u。

Slanted 字体

本组的字体来源于直立罗马字体，只是对它进行斜率为 1/6 的倾斜；也就是说，这些字体向右倾斜了高度的六分之一。Slanted 字体有介于 8 到 12 pt 间的四种设计尺寸。字体样式标志为 sl，因此所有成员的文件名以 cmsl 开头，后接的 8, 9, 10 或 12 表示设计尺寸。在 NFSS 中这些字体的形状定义为 sl。

cmsl10

ABCDEFGHIJKLMNOPQRSTUVWXYZ ÆŒ Ø @# \$ % &
 abcdefghijklmnopqrstuvwxyz ff fi fl ffi ffi æœøßij 0123456789
 ΓΔΘΛΞΠΣΥΦΨΩ`´˘˙˚˛˜˝˞˟ˠˡˢˣˤ˥˦˧˨˩˪˫ˬ˭ˮ˯˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿?ii'""- - —*+/() []

斜体

在斜体字体组中的字体在基本形式上不同于直立和 slanted 罗马字体。它的倾斜值为 1/4，要比 slanted 字体斜得更大。形状属性的标志为 it，而字体样式标志为 ti，表示文本斜体 (text italic)。因此字体文件的名称开头为 cmti，后接设计尺寸值。有从 7 到 12 pt 之间的五种尺寸。（数学字符字体属于另外一种斜体族，样式标志为 mi，代表数学斜体 (math italic)。）

cmti10

E.3.2 Sans serif 族

所有 sans serif 字体在其名称中都有样式标志 *ss*, 因此其名称开头为 *cmss*. 这也是其 NFSS 族的名称。

cmss 族, OT1 编码			
形状 =	n	sl	—
序列			
m	cmss8 cmss9 cmss10 cmss12 cmss17	cmssi8 cmssi9 cmssi10 cmssi12 cmssi17	
bx	cmssbx10		
sbc	cmssdc10		
—			cmssq8 cmssqi8 cminch

直立 sans serif 字体

介于 8 到 17 pt 之间有五种设计尺寸的 sans serif 字体可以使用, 名称分别为 *cmss5* ... *cmss17*. 还有一种字体 *cmssq8*, 称为 *sans serif quotation*, 其中的小写字母比通常情形按比例放大了一些。这种字体并不是按标准 NFSS 属性框架进行分类的。

cmss10

ABCDEFGHIJKLMNOPQRSTUVWXYZ ÆŒ Ø @#\$\$%&
 abcdefghijklmnopqrstuvwxyz ff fi fl ffi ffl æœøßij 0123456789
 ΓΔΘΛΞΠΣΥΦΨΩ`´ˆˇ˘˙˚˛˜˝˞˟ˠˡˢˣˤ˥˦˧˨˩˪˫ˬ˭ˮ˯˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿

cmssq8

ABCDEFGHIJKLMNOPQRSTUVWXYZ ÆŒ Ø @#\$\$%&
 abcdefghijklmnopqrstuvwxyz ff fi fl ffi ffl æœøßij 0123456789
 ΓΔΘΛΞΠΣΥΦΨΩ`´ˆˇ˘˙˚˛˜˝˞˟ˠˡˢˣˤ˥˦˧˨˩˪˫ˬ˭ˮ˯˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿

Slanted sans serif 字体

Slanted sans serif 字体是与同尺寸和编号的直立字体对应的。样式标志为 *ssi* 和 *ssqi*, 其中的 *i* 代表倾斜 (inclined), 而不是斜体 (italic). 正是因为如此, 它们被赋予属性 *sl*. 倾斜大致为 1/5. 另外, *quotation* 字体 *cmssqi8* 并不包含在 NFSS 框架中。

cmssi10

ABCDEFGHIJKLMNOPQRSTUVWXYZ ÆŒ Ø @#\$\$%&
 abcdefghijklmnopqrstuvwxyz ff fi fl ffi ffl æœøßij 0123456789

ΓΔΘΛΞΠΣΤΦΨΩ`´˘˙˚˛˜˝˞˟ˠˡˢˣˤ˥˦˧˨˩˪˫ˬ˭ˮ˯˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿?[]'""- - —*+/()[]

cmssqi8

ABCDEFGHIJKLMNOPQRSTUVWXYZ ÆŒ Ø @#\$%&
 abcdefghijklmnopqrstuvwxyz ff fi fl ffi ffl æœøðŷ 0123456789
 ΓΔΘΛΞΠΣΤΦΨΩ`´˘˙˚˛˜˝˞˟ˠˡˢˣˤ˥˦˧˨˩˪˫ˬ˭ˮ˯˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿?[]'""- - —*+/()[]

Sans serif 黑体

只有设计尺寸为 10 pt 的 sans serif 黑体。其中一个名为 cmssbx10, 表示 sans serif bold extended, 其与 cmss10 的关系就如同 cmbx10 与 cmr10 的关系。另外一种黑体是 cmssdc10, 表示 sans serif demibold condensed.

cmssbx10

**ABCDEFGHIJKLMNOPQRSTUVWXYZ ÆŒ Ø @#\$%&
 abcdefghijklmnopqrstuvwxyz ff fi fl ffi ffl æœøðŷ 0123456789
 ΓΔΘΛΞΠΣΤΦΨΩ`´˘˙˚˛˜˝˞˟ˠˡˢˣˤ˥˦˧˨˩˪˫ˬ˭ˮ˯˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿?[]'""- - —*+/()[]**

cmssdc10

**ABCDEFGHIJKLMNOPQRSTUVWXYZ ÆŒ Ø @#\$%&
 abcdefghijklmnopqrstuvwxyz ff fi fl ffi ffl æœøðŷ 0123456789
 ΓΔΘΛΞΠΣΤΦΨΩ`´˘˙˚˛˜˝˞˟ˠˡˢˣˤ˥˦˧˨˩˪˫ˬ˭ˮ˯˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿˰˱˲˳˴˵˶˷˸˹˺˻˼˽˾˿?[]'""- - —*+/()[]**

cminch 字体

在 cminch 字体中只有大写字母和 0,1,...,9 的数字, 而再没有其他符号。其中的字符高度均为一英寸。字体的名称是一个例外, 因为这时并没有用数字表示设计尺寸。对于这种情形, inch 既表示字体样式, 也表示尺寸。在 NFSS 中也没有这种字体的分类。

cminch

A B C D

123456

E.3.3 打字机字体

对于固定字体，每个字符都具有同样的宽度。甚至单词之间的距离也等于字体的宽度，而并不会进行伸展或收缩。然而，同所有 TeX 字体一样，单词间距总量并不与输入的空格数多少有关。标准的固定字体，都有一个字符表示打字机样式，它们是：

形状 = n sc sl it —

序列

m	cmtt8 cmtt9 cmtt10 cmtt12	cmtcsc10	cmsl10	cmitt10
---	------------------------------------	----------	--------	---------

cmtex8
cmtex9
cmtex10
cmvtt10

直立打字机字体

直立打字机字体有介于 8 到 12 pt 之间的四种设计尺寸。样式标志 `tt` 表示打字机类型 (typewriter type), 因此文件基本名为 `cmtt8 ... cmtt12`.

cmtt10

ABCDEFGHIJKLMNOPQRSTUVWXYZ AED @#\$%& ΓΔΘΑΞΠΕΤΦΨΩ
abcdefghijklmnopqrstuvwxyz æøß ßj;jç ()[]{}\\|/↑↓
0123456789 -+*<=> .,:;!?'~`-~_~^~^~^~^

大写与小大写

打字机大写与小大写字体 `cmtcsc10` 生成 10 pt 打字机字体的大写字母, 而小写字母则为 8 pt 的打字机大写字母。

cmtcsc10

ABCDEFGHIJKLMNOPQRSTUVWXYZ EEF @#\$%& ΓΔΘΛΞΠΕΤΦΨΩ
 ABCDEFGHIJKLMNOPQRSTUVWXYZ EEΘ SSIJ; () [] { } \ | / + ↑
 0123456789 -+*<=> _ . , : ; ! ? ~ ^ ` ~ ~ ~ ~ ~

E.5 数学与符号字体

数学与符号字体就是由数学公式中需要的特殊字符组成。另外，在 \LaTeX 的 `picture` 环境中也需要用到特殊符号，例如斜线、箭头，都可以在 \LaTeX 符号字体中找到。最后提一下，一些特殊记号字体也包括其中。

E.5.1 数学字体族

数学字体所具有的编码框架与普通文本中的字体没有任何关联。正是出于这个原因，每一种字体都具有自己的编码和族定义。

数学字体族			
编码 =	OML	OMS	OMX
族 =	cmm	cmsy	cmex
形状 =	it	n	n
序列			
m	<div> cmmi5 cmmi6 cmmi7 cmmi8 cmmi9 cmmi10 cmmi12 </div>	<div> cmsy5 cmsy6 cmsy7 cmsy8 cmsy9 cmsy10 </div>	<div>cmex10</div>
b	<div>cmmib10</div>	<div>cmsyb10</div>	

数学文本字体

这些字体由大写和小写的拉丁字母与希腊字母组成，还包括其他一些符号。由于在公式中有各种名称的斜体，而这些是基本的斜体字体，因此其字体样式标志为 `mi`，表示数学斜体 (*mathematical italic*)，以与通常的斜体区分开。字体有介于 5 到 12 pt 之间的七种设计尺寸。对于 10 pt 的设计尺寸，还有一种数学斜黑字体 `mib10`。

`cmmi10`

$\Gamma\Delta\Theta\Lambda\Xi\Pi\Sigma\Upsilon\Phi\Psi\Omega\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\pi\rho\sigma\tau\upsilon\phi\chi\psi\omega\epsilon\vartheta\varpi\rho\varsigma\varphi\leftarrow\rightarrow\rightarrow^{\circ}\triangleright\triangleleft$
0123456789.,</>* ∂ ABCDEFGHIJKLMN O P Q R S T U V W X Y Z
bq#~\~lab c d e f g h i j k l m n o p q r s t u v w x y z i j p ~ ~

`cmmib10`

$\Gamma\Delta\Theta\Lambda\Xi\Pi\Sigma\Upsilon\Phi\Psi\Omega\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\pi\rho\sigma\tau\upsilon\phi\chi\psi\omega\epsilon\vartheta\varpi\rho\varsigma\varphi\leftarrow\rightarrow\rightarrow^{\circ}\triangleright\triangleleft$
0123456789.,</>* ∂ ABCDEFGHIJKLMN O P Q R S T U V W X Y Z
bq#~\~lab c d e f g h i j k l m n o p q r s t u v w x y z i j p ~ ~

数学符号

这些符号字体给出的是公式中其他数学符号，但尺寸可以变化的除外。字体样式代

LaTeX 的 lasy 字体

作为对数学符号字体 `cmsy` 的推广, LaTeX 提供了从 5 到 10 pt 六种设计尺寸的其他一些符号。这些字体文件的基本名为 `lasy5 ... lasy10`。其中每个包含 16 个符号:

$\diamond, \wedge, \triangleleft, \triangle, \triangleright, \triangleright, \cup, \boxtimes, \square, \sim, \leadsto, \square, \square$ 。

在 LaTeX 2_ε 中, 只有用了 `latexsym` 宏包, 这些字体才有定义。

生成图形的字体

在 LaTeX 的 `picture` 环境中, 所使用的特殊图形元素被保存在名称分别为 `line10`, `lcircle10`, `linew10` 和 `lcirclew10` 的字体文件中。前两个用来生成 `\thinlines` (6.5.1 节) 起作用时的直线、卵形线和圆。斜线和箭头 (6.4.3 和 6.4.4 节) 可以在 `line10` 中找到, 而圆与卵形线片断 (6.4.5 和 6.4.6 节) 可以在 `lcircle10` 中找到。后面那对字体名称中增加了 `w`, 表示粗线, 这是用在 `\thicklines` 起作用的情形中。

记号字体

在记号字体 `logo8`, `logo9`, `logo10`, `logos110` 和 `logobf10` 中只有七个字母 A, E, F, M, N, O, T, 用来生成记号

METAFONT METAFONT METAFONT

E.6 字体文件

E.6.1 基本名

在每个 TeX 的实现版本中, 都应该包含列在 E.3 至 E.5 节中各种不同设计尺寸的多达 75 种标准字体。这些字体的基本名汇总如下:

<code>cmr5</code>	<code>cmti9</code>	<code>cmssq8</code>	<code>cmu10</code>	<code>cmmi5</code>
<code>cmr6</code>	<code>cmti10</code>	<code>cmss8</code>	<code>cmff10</code>	<code>cmmi6</code>
<code>cmr7</code>	<code>cmti12</code>	<code>cmss9</code>	<code>cmfi10</code>	<code>cmmi7</code>
<code>cmr8</code>	<code>cmbx5</code>	<code>cmss10</code>	<code>cmdunh10</code>	<code>cmmi8</code>
<code>cmr9</code>	<code>cmbx6</code>	<code>cmss12</code>	<code>cmitt8</code>	<code>cmmi9</code>
<code>cmr10</code>	<code>cmbx7</code>	<code>cmss17</code>	<code>cmitt9</code>	<code>cmmi10</code>
<code>cmr12</code>	<code>cmbx8</code>	<code>cmssqi8</code>	<code>cmitt10</code>	<code>cmmi12</code>
<code>cmr17</code>	<code>cmbx9</code>	<code>cmssi8</code>	<code>cmitt12</code>	<code>cmmib10</code>
<code>cmcsc10</code>	<code>cmbx10</code>	<code>cmssi9</code>	<code>cmtcsc10</code>	<code>cmsy5</code>
<code>cmsl8</code>	<code>cmbx12</code>	<code>cmssi10</code>	<code>cmslitt10</code>	<code>cmsy6</code>
<code>cmsl9</code>	<code>cmb10</code>	<code>cmssi12</code>	<code>cmitt10</code>	<code>cmsy7</code>
<code>cmsl10</code>	<code>cmfib8</code>	<code>cmssi17</code>	<code>cmtex8</code>	<code>cmsy8</code>
<code>cmsl12</code>	<code>cmbxsl10</code>	<code>cmssdc10</code>	<code>cmtex9</code>	<code>cmsy9</code>
<code>cmti7</code>	<code>cmbxti10</code>	<code>cmssbx10</code>	<code>cmtex10</code>	<code>cmsy10</code>
<code>cmti8</code>	<code>cminch</code>	<code>cmvtt10</code>	<code>cmex10</code>	<code>cmbsy10</code>

AMS 提供了黑体数学斜体 `cmmib` 以及黑体符号字体 `cmbsy`, 设计尺寸为 5–9 pt。安装 LaTeX 2_ε 时, 就假定这些字体是存在的, 虽然如果没有它们的话, 也没有什么坏处。

```

cmmib5  cmmib6  cmmib7  cmmib8  cmmib9
cmbsy5  cmbsy6  cmbsy7  cmbsy8  cmbsy9

```

除了标准 TeX 字体外, 还有一些特殊的记号字体以及 L^AT_EX 字体:

```

lasy5  lasy8  lasyb10  linewidth10  logo8  logobf10
lasy6  lasy9  line10    lcirclew10  logo9
lasy7  lasy10 lcircle10 logosl10  logo10

```

在基本名末尾的数字表示设计尺寸的点数。对这些字体中的每一个, 都存在一个以此为基本名、扩展名为 .tfm 的文件, 例如 cmr10.tfm。在 TeX 和 L^AT_EX 处理文本文件的过程中, 只用到这些 .tfm 文件。 .tfm 文件由尺寸信息组成, 还有字体中的其他字符以及符号, 见 325 页上的说明, 但并没有说明符号到底是什么样子的。

只有要用驱动程序把输出送到打印机上时, 才会需要知道如何构造这些字母和符号。这些构造信息存贮在其他的文件中, 除各种基本尺寸外, 还可以进行不同的放大。

E.6.2 字体放大

TeX 字体通常的放大幅度是 1.2 的幂次以及 $\sqrt{1.2}$ 。在 L^AT_EX 中, 字体尺寸的选择是利用 4.1.2 节的命令实现的, 这些命令会检测是否有要求尺寸的相应字体, 如果没有, 就会对另外一种设计尺寸进行适当的放大。L^AT_EX 2.09 是从 lfonts.tex 文件中接受字体信息的, 而 L^AT_EX 2_ε 用的则是字体定义文件 (.fd, 8.5.8 节)。对字体的放大也可以用下述命令来选择:

```
\newfont{\字体命令}{基本名 scaled 放缩因子}
```

这样就会把 \字体命令 定义成激活放大 放缩因子/1000 倍的基本名字体的命令。也就是说 放缩因子 是等于放大倍数 1000 倍的整数。如果用的就是设计尺寸, 那该数就是 1000; 对第一个放大幅度 $\sqrt{1.2}$, 其值为 1095; 然后接下来就是 1200, 1440, ... TeX 命令 \magstepn 可以生成 1.2^n , 而 \magstephalf 就是 $\sqrt{1.2}$ 。

10 pt 的字体被放大 1.2 倍并不与设计尺寸为 12 pt 的字体相同, 虽然这种差别是很难发现的, 例如,

12pt unscaled 10pt scaled by 1200

当放大倍数是 1.2^3 , 即放缩因子为 1728 时差别就比较明显了:

17pt unscaled 10pt scaled by 1728

放大的字体虽然可能具有与原始字体同样的高度, 但字符显得更黑, 并且要宽一些。这是因为小设计尺寸字体在字母高度和宽度以及线粗等方面的关系与大尺寸字体的不同。然而, 当把它们放大到同样高度时, 所有其他度量都同比例地放大。当字体以设计尺寸显示时, 它是最好看的。

参考文献

- [1] Adobe Systems. PostScript Language Reference Manual, 2nd edition. Addison-Wesley: 1990
- [2] 丁卫星, 赖天树. \LaTeX 排版软件实用教程. 中国科学技术大学出版社: 1993
- [3] 常庚哲. 曲面的数学. 湖南教育出版社: 1995
- [4] Goosens M, Mittelbach F and Samarin A. The \LaTeX Companion. Addison-Wesley: 1994
- [5] Goosens M, Rahtz S and Mittelbach F. The \LaTeX Graphics Companion—Illustrating Documents with \TeX and PostScript. Addison-Wesley: 1997
- [6] Grätzer G. Math into \LaTeX , Birk Häuser: 1996
- [7] 郭力, 张林波等. CCT 中外文激光照排系统. 海洋出版社: 1993
- [8] Knuth D E. The \TeX book. Computers and Typesetting. Vol. A. Addison-Wesley: 1984
- [9] Knuth D E. The METAFONTbook. Computers and Typesetting. Vol. C. Addison-Wesley: 1986
- [10] Kopka H and Daly P W. A Guide to $\text{\LaTeX} 2_{\epsilon}$ — Document Preparation for Beginners and Advanced Users. 2nd edition. Addison-Wesley: 1997
- [11] Lamport L. \LaTeX — A Document Preparation System. Addison-Wesley: 1985
- [12] Lamport L. \LaTeX — A Document Preparation System. 2nd edn. for $\text{\LaTeX} 2_{\epsilon}$. Addison-Wesley: 1994
- [13] Reid G C. Thinking in PostScript. Addison-Wesley: 1990
- [14] Salomon D. The Advanced \TeX book. Springer-Verlag, New York: 1995

索引

(按 ASCII 码排序)

- !(MakeIndex), 189
- \!, 118
- !', 19, 239
- ", 18
- "(MakeIndex), 190
- \\", 19, 239
- \", 190
- #, 15, 164, 165, 172, 173, 175, 177, 179
- ##, 176
- \#, 19, 239, 314
- \$, 15, 22, 92, 313
- \\$, 15, 19, 239, 303
- %, 15, 18, 86, 86
- \%, 19, 239
- &, 15, 21, 23, 24, 28, 30, 104, 105, 143, 145, 149
- \&, 19, 239
- ', 18
- '', 18
- \'(`重音), 19, 239
- \'(制表), 76
- (, 93, 102, 126
- \(, 22, 92, 306, 313
-), 93, 102, 126
- \), 92, 306
- \+(制表), 75, 309
- \,, 48, 96, 118
- , 19
- \-, 321
- \-(断词), 54, 321
- \-(制表), 75, 76, 309
- , 19
- , 19
- \.(`重音), 19, 239
- /, 93
- \/, 48
- \:, 103, 104, 118
- \;, 103, 104, 118
- \<, 305, 309
- \<(制表), 75
- \=(`重音), 19, 239
- \=(制表), 74, 75, 76, 78, 309
- \>(制表), 74, 75, 76, 77, 309
- ?', 19, 239
- @(MakeIndex), 189–191
- \@, 48
- @- 表达式, 21, 142, 143, 144, 146
- \@date, 275
- \@ifnextchar, 275
- \@ifstar, 275
- \@ifundefined, 275
- \@latexerr, 266
- \@namedef, 275
- \@nameuse, 275
- \@warning, 266
- [, 12, 15, 93, 102, 314
- \[, 22, 92, 306, 313
- \\, 42, 50, 53, 62, 63, 74, 108, 132, 143–146, 150, 305, 309, 313, 321
- *, 50
-], 12, 15, 21, 102
- \], 22, 92, 101
- ~, 15, 93, 94, 313
- \^, 19, 239
- \^(`重音), 19, 239
- _ , 15, 19, 93, 94, 98, 239, 304, 313
- _, 19, 239, 304
- ' , 18

- '', 18
- \(' 重音), 19, 239
- \'(制表), 76
- {, 15
- \{, 19, 93, 239
 - |, 240
- \|, 102
 - | (表格), 143
 - }, 12, 313, 314
- \}, 19, 102, 239
 - | (MakeIndex), 191
 - ~, 15, 47
- \~(重音), 19, 239
 - 10pt选项, 33, 57
 - 11pt选项, 33, 57
 - 12pt选项, 33, 57
- _, 参见 空白
- \a' (制表), 76
 - a4paper选项, 34, 273
 - a5paper选项, 34
- \a= (制表), 76
- \a' (制表), 76
- \AA, 19, 239
- \aa, 19, 239
- \abovedisplayshortskip, 121
- \abovedisplayskip, 121
 - abstract环境, 43
 - DC 字体, 200
- \RequirePackage, 310
- \acute, 99, 244
- \addcontentsline, 46, 47, 155, 315, 323
- \address, 212, 213
- \addtime(slides), 211
- \addtocontents, 21, 155, 323
- \addtocounter, 17, 85, 88, 89, 160, 170, 308
- \addtolength, 161, 177, 179
- \addvspace, 162
- \AE, 19, 202, 239
- \ae, 19, 239
 - afterpage.sty(宏包), 206
- \aleph, 98, 240
 - align环境, 113
 - align*环境, 113
 - alignat环境, 114
 - aligned环境, 115
 - \allowdisplaybreaks, 115
 - alltt环境, 86, 204
 - alltt.sty(宏包), 86, 204
 - \Alph, 66
 - \Alpha, 66
 - Alpha, 39
 - alpha, 39
 - \alpha, 240
 - \amalg, 242
 - amscd宏包, 116
 - amslatex, 92, 112, 113, 207
 - \and, 42
 - \And, 242
 - \angle, 244
 - appendix, 45
 - \appendix, 45
 - \appendixname, 233
 - \approx, 240
 - \approxeq, 241
 - arabic, 39
 - \arabic, 21, 66, 69, 71, 72, 88, 160, 161, 171, 172, 179
 - \arccos, 244
 - \arcsin, 244
 - \arctan, 244
 - \arg, 244
 - array环境, 102-105, 107, 108, 119, 120, 122, 123, 142, 144, 307, 308, 313, 314, 316, 317
 - array.sty(宏包), 206
 - \arraycolsep, 121, 144
 - \arrayrulewidth, 144
 - \arraystretch, 144
 - article类, 33, 34-36, 43-46, 68, 87, 101, 155, 166, 187, 204, 218, 224, 228, 264, 268, 270, 292, 294, 302
 - article.cls, 269, 273, 318

-
- article.sty, 264, 274, 318
 - \ast, 242
 - \asympt, 240
 - \AtBeginDocument, 270
 - \AtEndDocument, 270
 - \AtEndOfClass, 270
 - \AtEndOfPackage, 270
 - \atop, 106, 107, 119, 120, 165
 - \author, 22, 42, 43
 - .aux文件, 186
 - axiom, 73
 - \b(重音), 19, 239
 - b5paper选项, 34
 - babel多语种 L^AT_EX, 207
 - \backepsilon, 241
 - \backmatter, 45, 182
 - \backprime, 244
 - \backsim, 241
 - \backsimeq, 241
 - \backslash, 102, 243
 - \bar, 99, 244
 - \barwedge, 242
 - \baselineskip, 39, 58, 193
 - \baselinestretch, 39, 58, 59
 - \Bbbk, 244
 - \because, 241
 - \begin, 16, 21, 306, 307
 - \begin{abstract}, 43
 - \begin{align}, 113
 - \begin{alignat}, 114
 - \begin{aligned}, 115
 - \begin{appendix}, 45
 - \begin{array}, 102, 104, 105, 108, 126, 142
 - \begin{belowdisplayshortskip}, 122
 - \begin{belowdisplayskip}, 121
 - \begin{center}, 62
 - \begin{description}, 63, 64
 - \begin{displaymath}, 92, 122, 123
 - \begin{document}, 12, 22, 225, 228, 270, 273, 292, 294, 295, 298–302, 306, 308, 322
 - \begin{enumerate}, 63, 64, 65
 - \begin{eqnarray}, 93, 108, 110
 - \begin{eqnarray*}, 93, 108, 109, 110
 - \begin{equation}, 92, 100, 122, 123
 - \begin{equation*}, 101
 - \begin{figure}, 152
 - \begin{figure*}, 152
 - \begin{filecontents}, 274
 - \begin{filecontents*}, 274
 - \begin{flalign}, 114
 - \begin{flushleft}, 62
 - \begin{flushright}, 62
 - \begin{fussypar}, 55
 - \begin{gather}, 112
 - \begin{gather*}, 112
 - \begin{gathered}, 115
 - \begin{itemize}, 63, 63
 - \begin{letter}, 21, 212
 - \begin{list}, 68, 71, 72, 177, 178
 - \begin{lrbox}, 79
 - \begin{math}, 92
 - \begin{minipage}, 79, 80–82, 82
 - \begin{multline}, 112
 - \begin{multline*}, 113
 - \begin{note}(slides), 210
 - \begin{overlay}(slides), 210
 - \begin{picture}, 126, 130, 131, 133–136, 138–140, 156, 157
 - \begin{quotation}, 62
 - \begin{quote}, 16, 62, 170–172
 - \begin{slide}(slides), 209
 - \begin{sloppypar}, 55, 172
 - \begin{tabbing}, 74, 75, 76
 - \begin{table}, 152, 156, 157
 - \begin{table*}, 152
 - \begin{tabular}, 142, 145–147, 149, 150, 156, 157
 - \begin{tabular*}, 142
 - \begin{thebibliography}, 67, 186
 - \begin{theindex}, 187
 - \begin{theorem}, 74

-
- `\begin{titlepage}`, 42
 - `\begin{trivlist}`, 72
 - `\begin{verbatim}`, 85
 - `\begin{verbatim*}`, 85
 - `\begin{verse}`, 63
 - `\beta`, 105, 106, 110, 240
 - `\beth`, 240
 - `\between`, 241
 - `\bezier`, 134
 - `bezier.sty`(宏包), 134
 - `\bf`, 4, 60, 61, 72, 103, 112, 191, 204, 205, 233
 - `\bfdefault`, 195
 - `\bfseries`, 16, 60, 61, 71, 72, 147, 149, 150, 157, 158, 195, 196, 204, 306
 - `\bibitem`, 21, 67, 68, 186, 187
 - `\bibname`, 233
 - `BIBTeX`, 187, 188
 - `\Big`, 120
 - `\big`, 120
 - `\bigcap`, 245
 - `\bigcirc`, 242
 - `\bigcup`, 245
 - `\Bigg`, 120
 - `\bigg`, 120
 - `\Biggl`, 121
 - `\biggl`, 121
 - `\Biggm`, 121
 - `\biggm`, 121
 - `\Biggr`, 121
 - `\biggr`, 121
 - `\Bigl`, 121
 - `\bigl`, 121
 - `\Bigm`, 121
 - `\bigm`, 121
 - `\bigodot`, 245
 - `\bigoplus`, 245
 - `\bigotimes`, 245
 - `\Bigr`, 121
 - `\bigr`, 121
 - `\bigskip`, 51
 - `\bigskipamount`, 51
 - `\bigsqcup`, 245
 - `\bigstar`, 244
 - `\bigtriangledown`, 242
 - `\bigtriangleup`, 242
 - `\biguplus`, 245
 - `\bigvee`, 245
 - `\bigwedge`, 245
 - `\binom`, 107
 - `\blacklozenge`, 244
 - `\blacksquare`, 244
 - `\blacktriangle`, 244
 - `\blacktriangledown`, 244
 - `\blacktriangleleft`, 241
 - `\blacktriangleright`, 241
 - `\bmod`, 99
 - `\boldmath`, 111, 112, 196, 320
 - book 类, 33, 34, 37, 41, 43-46, 53, 68, 87, 155, 161, 187, 204, 224, 228, 264, 302
 - `book.sty`, 264
 - `\boolean`, 169, 275, 277, 278
 - `\bot`, 243
 - `\botfigrule`, 154
 - `\bottomfraction`, 154
 - `bottomnumber`, 154
 - `\bowtie`, 240
 - `\Box`, 244
 - `\boxdot`, 242
 - `\boxed`, 123
 - `\boxminus`, 242
 - `\boxplus`, 242
 - `\boxtimes`, 242
 - `bp`(大点), 17
 - `\breve`, 244
 - `\bullet`, 242
 - `\Bumpeq`, 241
 - `\bumpeq`, 241
 - `\c`(重音), 19, 239
 - `\cal`, 4, 98
 - `\Cap`, 242
 - `\cap`, 98, 242

-
- \caption, 21, 155, 156, 160, 185, 315, 323
 - cc(cicero), 17
 - \cc, 213
 - Cc2TeX, 218
 - \ccdp, 232
 - \ccht, 232
 - \ccname, 213
 - \ccnospace, 232
 - CCT
 - cct.dat, 229
 - cct.exe, 225, 236
 - cctinit.exe, 221
 - 处理流程, 224
 - 配置, 221
 - True Type 字体, 237
 - 文件组成, 220
 - 系统网址, 220
 - 源文档结构, 228
 - cctart类, 33
 - cctbook类, 33
 - \ccwd, 232
 - CD环境, 116
 - cdiva.exe, 237
 - \cdot, 100, 103–105, 107, 109, 242
 - \cdots, 97
 - center环境, 62, 73, 126, 144
 - \centerdot, 242
 - \centering, 62, 73
 - \centerline, 62, 298
 - chapter计数器, 74, 159, 161, 166
 - \chapter, 20, 39, 43, 44, 45
 - \chapter*, 43
 - \chaptername, 233
 - \check, 244
 - \CheckCommand, 271
 - \CheckCommand*, 271
 - \chi, 240
 - ChiLaTeX, 218
 - ChTeX, 217
 - ChiTeX, 218
 - \chnno, 232
 - \chnno@one, 233
 - \chnno@two, 233
 - \chntoday, 232
 - \choose, 106, 119, 165
 - \circ, 242
 - \circeq, 241
 - \circle, 130, 133, 135, 138, 306
 - \circle*, 130, 132, 140
 - \circledS, 244
 - \circledast, 242
 - \circledcirc, 242
 - \circleddash, 242
 - \cite, 67, 68, 186, 187
 - CJK(宏包), 218
 - \ClassError, 271
 - \ClassInfo, 272
 - \ClassWarning, 272
 - \ClassWarningNoLine, 272
 - \cleardoublepage, 52, 153, 309
 - \clearpage, 52, 77, 153, 181, 309, 316
 - \cline, 143, 148, 150
 - clock选项(slides), 210
 - \closing, 213
 - \clubpenalty, 53
 - \clubsuit, 244
 - cm(厘米), 17
 - \columnsep, 34, 35, 42
 - \columnseprule, 34, 35
 - \columnwidth, 41, 155
 - \complement, 244
 - Comprehensive TeX Archive Network, 参见 CTAN
 - Computer Modern Fonts, 参见 计算机现代字体
 - \cong, 240
 - \contentsline, 47
 - \contentsname, 233
 - \coprod, 245
 - \copyright, 19, 239
 - \copyrightspace, 204
 - \cos, 244

-
- \cosh, 244
 - \cot, 244
 - \coth, 244
 - \CS, 232
 - \csc, 244
 - CTAN 文件服务器, 207
 - \Cup, 242
 - \cup, 242
 - \curlyeqprec, 241
 - \curlyeqsucc, 241
 - \curlyvee, 242
 - \curlywedge, 242
 - \CurrentOption, 269, 270
 - Cyrillic 字体, 191
 - \d(重音), 19, 239
 - \dag, 19, 244
 - \dagger, 242
 - \daleth, 240
 - \dashbox, 127, 129, 135
 - \dashv, 240
 - \date, 42, 43
 - \date(书信), 213
 - \day, 20
 - \dblfigrule, 154
 - \dblfloatpagefraction, 154
 - \dblfloatsep, 154
 - \dbltextfloatsep, 154
 - \dbltopfraction, 154
 - dbltopnumber, 154
 - dcolumn.sty(宏包), 206
 - dd(didôt), 17
 - \ddag, 19, 244
 - \ddagger, 242
 - \ddddot, 244
 - \dddot, 244
 - \ddot, 244
 - \ddots, 97
 - \DeclareErrorFont, 199, 312
 - \DeclareFixedFont, 61, 196, 325
 - \DeclareFontEncoding, 198, 200, 201
 - \DeclareFontEncodingDefaults, 199
 - \DeclareFontFamily, 199, 200, 312
 - \DeclareFontShape, 199, 200, 312
 - \DeclareFontSubstitution, 199
 - \DeclareInputMath, 202
 - \DeclareInputText, 202
 - \DeclareMathAccent, 312
 - \DeclareMathAlphabet, 197, 306, 312
 - \DeclareMathSizes, 198
 - \DeclareMathSymbol, 197
 - \DeclareMathVersion, 197, 311, 312
 - \DeclareOldFontCommand, 196
 - \DeclareOption, 269, 278, 310
 - \DeclareOption*, 269, 270
 - \DeclareRobustCommand, 268, 270
 - \DeclareRobustCommand*, 271
 - \DeclareSymbolFont, 197, 312, 313
 - \DeclareSymbolFontAlphabet, 197, 313
 - \DeclareTextAccent, 201
 - \DeclareTextAccentDefault, 201
 - \DeclareTextCommand, 201, 306
 - \DeclareTextCommandDefault, 201
 - \DeclareTextComposite, 201, 311
 - \DeclareTextCompositeCommand, 201
 - \DeclareTextSymbol, 201
 - \DeclareTextSymbolDefault, 201
 - \def, 266, 267, 271, 276
 - \deg, 244
 - delarray.sty(宏包), 206
 - \DeleteShortVerb, 205
 - \Delta, 240
 - \delta, 240
 - \depth, 78, 82
 - description环境, 63, 64, 68, 73, 178, 189, 309
 - \det, 99, 244
 - \dfrac, 94
 - \diagdown, 244
 - \diagup, 244
 - \Diamond, 244
 - \diamond, 242
 - \diamondsuit, 244

- `\digamma`, 240
- `\dim`, 244
- `\discretionary`, 54
 - displaymath环境, 92
- `\displaystyle`, 118, 119, 120, 122
- `\div`, 242
- `\divideontimes`, 242
 - docstrip.tex, 204
- `\documentclass`, 4, 6, 8, 9, 11, 12, 33, 35, 36, 41, 57, 58, 182, 194, 204, 209, 211, 212, 218, 224, 225, 228, 264, 268, 270, 273, 274, 277, 279, 292, 294, 297, 299, 301, 302, 306, 307, 310–312, 319
- `\documentstyle`, 4, 12, 33, 36, 58, 134, 188, 224, 264, 274, 297, 307, 310, 311
- `\dot`, 244
- `\doteq`, 240
- `\doteqdot`, 241
- `\dotfill`, 49, 77, 78, 104
- `\dotplus`, 242
- `\doublebarwedge`, 242
- `\doublerulesep`, 144
- `\Downarrow`, 243
- `\downarrow`, 102
- `\downarrow`, 102, 243
- `\downdownarrows`, 243
- `\downharpoonleft`, 243
- `\downharpoonright`, 243
 - draft选项, 35
 - .dvi文件, 13, 218
 - dvips, 209, 238
 - dvips32, 238
 - e.tex, 303
- `\edef`, 276
 - Edit(编辑程序), 7
 - EditPlus, 7
 - 彩色显示, 7
 - 配置, 7
 - 语法文件, 7
 - 自动补足, 7
- `\ell`, 243
- `\else`, 277
 - em, 17, 49
- `\em`, 16, 57, 58, 60, 192, 305, 320
- `\emph`, 57, 60, 63, 192, 195
 - empty页面样式, 37, 210, 212
- `\emptyset`, 243
 - emTeX
 - 递归搜索, 223
 - 环境变量, 222
 - 目录结构, 222
 - 内存布局, 224
- `\encl`, 213
- `\enclname`, 213
- `\encodingdefault`, 195
- `\end`, 16, 20, 306
- `\end{...}`, 参见 `\begin` 命令
- `\end{document}`, 9, 12, 22, 153, 182, 184, 209, 225, 228, 270, 273, 277, 292, 294, 299, 302, 315, 322
- `\endinput`, 277
- `\enlargethispage`, 53, 308
- `\enlargethispage*`, 53
- `\ensuremath`, 163, 164, 165, 307
 - enumerate环境, 63, 64–66, 68, 73, 185, 309
 - enumerate.sty(宏包), 206
 - enumn 计数器, 66, 159
 - epsfig.sty宏包, 215
- `\epsilon`, 240
- `\eqcirc`, 241
 - eqnarray环境, 93, 108, 184, 185, 309
 - eqnarray*环境, 93, 108, 184
- `\eqref`, 101
- `\eqslantgtr`, 241
- `\eqslantless`, 241
- `\equal`, 168
 - equation环境, 92, 101
 - equation计数器, 159
- `\equiv`, 240
 - errorcontextlines计数器, 298
- `\eta`, 240

- \eth, 244
- \evensidemargin, 40, 279
 - ex, 17
- \ExecuteOptions, 269, 278
 - executivepaper选项, 34
- \exists, 243
- \exp, 244
- \expandafter, 276
- \exscale.sty(宏包), 204
- \extracolsep, 142, 143
- \f@baselineskip, 198
- \f@encoding, 198
- \f@family, 198
- \f@series, 198
- \f@shape, 198
- \f@size, 198
- \fallingdotseq, 241
- \familydefault, 195
- \fbox, 77, 83–85, 89, 111, 122, 123
- \fbox(图形), 133
- \fboxrule, 84, 123
- \fboxsep, 84, 123, 133
- \fi, 86, 277
 - figure计数器, 159
 - figure环境, 47, 152, 155, 156, 160, 184, 185, 216, 307–309, 318
 - figure*环境, 152
- \figurename, 233
 - filecontents环境, 274, 319
 - filecontents*环境, 274
 - fileerr.dtx.sty(宏包), 206
- \fill, 18, 142, 143, 162, 172, 173
 - final选项, 35
- \Finv, 244
 - flalign环境, 114
- \flat, 244
 - fleqn选项, 34, 92, 121, 122
- \floatpagefraction, 154
- \floatsep, 154
- \flushbottom, 41, 53
 - flushleft环境, 62, 73
 - flushright环境, 62, 73
- \flushright, 91
- \fnsymbol, 88, 160, 166
- \font, 167
- \fontdimen, 167
 - fontenc.sty(宏包), 200, 204
- \fontencoding, 167, 192, 194, 195
- \fontfamily, 167, 192, 193, 194
- \fontseries, 192, 193, 194
- \fontshape, 192, 193–195
- \fontsize, 192, 193, 194
 - fontsmpl.sty(宏包), 206
- footnote计数器, 85, 88, 159, 166
- \footnote, 87, 88, 89
- \footnotemark, 88, 89
- \footnoterule, 89
- \footnotesep, 89
- \footnotesize, 58, 194, 315, 316
- \footnotetext, 85, 88, 89
- \footskip, 40, 41, 278
- \forall, 243
- \frame, 133, 137
- \framebox, 5, 77, 78, 79, 82, 84, 85
- \framebox(图画), 127, 128, 133, 135, 137
- \frenchspacing, 48
- \frontmatter, 45, 182
- \frown, 240
 - ftnright.sty(宏包), 206
- fullpage宏包, 277
- \fussy, 55
 - fussypar环境, 55
- \Game, 244
- \Gamma, 240
- \gamma, 240
 - gather环境, 112
 - gather*环境, 112
 - gathered环境, 115
- \gcd, 99, 244
- \gdef, 266, 276
- \ge, 240
- \genfrac, 94

-
- `\geq`, 240
 - `\geqslant`, 241
 - `\gets`, 243
 - `\gg`, 240
 - `\ggg`, 241
 - `\gimel`, 240
 - `\glossary`, 177, 189, 310, 316
 - `\glossaryentry`, 189
 - `\gnapprox`, 241
 - `\gneq`, 241
 - `\gneqq`, 241
 - `\gnsim`, 241
 - `\grave`, 244
 - `\gtrapprox`, 241
 - `\gtrdot`, 241
 - `\gtreqless`, 241
 - `\gtreqqless`, 241
 - `\gtrless`, 241
 - `\gtrsim`, 241
 - `\gvertneqq`, 241
 - `\H("重音)`, 19, 239
 - `h.tex`, 303
 - `\hat`, 244
 - `\hbar`, 243
 - `\hbox`, 266
 - `\headheight`, 40, 278, 279
 - `headings`页面样式, 37, 43, 210, 278
 - `\headsep`, 40, 41
 - `\heartsuit`, 244
 - `\height`, 78, 82
 - `\hfill`, 49, 51, 55, 71, 77, 81, 84, 90, 91, 110, 111, 122, 156, 171, 177
 - `hhline.sty`(宏包), 206
 - `\hline`, 143, 146, 149
 - `\hom`, 244
 - `\hookleftarrow`, 243
 - `\hookrightarrow`, 243
 - `\hrulefill`, 49, 77, 82
 - `\hslash`, 244
 - `\hspace`, 48, 49, 75, 83, 102, 110, 143, 150, 151
 - `\hspace*`, 49, 75, 172
 - `\hss`, 298
 - `\Huge`, 58, 191, 194
 - `\huge`, 58, 194
 - `hyphen.tex`, 55
 - `\hyphenation`, 54, 55, 315, 316
 - `\i`, 19, 239
 - `\idotsint`, 96
 - `idx.tex`, 188, 203
 - `\if`, 277
 - `\ifcase`, 277
 - `\iff`, 243
 - `\iffalse`, 86
 - `\IfFileExists`, 272
 - `ifthen.sty`宏包, 167, 204, 266, 273
 - `\ifthenelse`, 167, 168, 169, 266, 275, 277, 278
 - `\iiiint`, 96
 - `\iiint`, 96
 - `\iint`, 96
 - `\Im`, 243
 - `\imath`, 243
 - `\in`, 240
 - `in(英寸)`, 17
 - `\include`, 159, 181, 182, 293, 301, 303, 304, 307
 - `\includegraphics`, 214
 - `\includegraphics*`, 214
 - `\includeonly`, 181, 182, 183, 211, 306
 - `\indent`, 51
 - `indentfirst.sty`(宏包), 206
 - `\index`, 177, 188, 189, 190, 203, 205, 234, 310, 316
 - `\indexentry`, 188
 - `\indexname`, 187, 233
 - `\indexspace`, 187
 - `\inf`, 99, 244
 - `\infty`, 243
 - `\initex`, 2
 - `\injl`, 244
 - `\input`, 173, 180, 181, 293, 301, 303, 304

- `inputenc.sty`(宏包), 202, 204
- `\InputIfFileExists`, 269, 272, 273
- `\int`, 245
- `\intercal`, 242
- `\intertext`, 114
- `\intertextsep`, 154
- `\invisible(slides)`, 210
- `\iota`, 240
- `\isodd`, 168
- ISO-Latin 编码, 202
- ISO-Latin1, 202
- `\it`, 4, 60, 191, 196
- `\itdefault`, 195
- `\item`, 63–66, 68, 69, 71, 72, 185, 187, 188–190, 300, 306–309
- `\item`选项, 64, 65, 72
- `\itemindent`, 70, 73
- `itemize`环境, 63, 64–66, 68, 73, 300
- `\itemsep`, 69, 71, 177–179
- `\itshape`, 48, 59, 148, 171, 173, 194–196, 306
- `\j`, 19, 239
- `\jmath`, 243
- `\Join`, 240
- `\jot`, 121
- `\kappa`, 240
- `\ker`, 244
- `\kill`, 75
- `\L`, 19, 239
- `\l`, 19, 239
- `\label`, 44, 101, 108, 158, 160, 177, 185, 186, 318, 319
- `\labelenumn`, 66
- `\labelitemn`, 66
- `\labelsep`, 70, 71, 177–179
- `\labelwidth`, 70, 71, 73, 177, 179
- `labl.st.tex`, 203
- `\Lambda`, 240
- `\lambda`, 240
- `landscape`选项, 34
- `\langle`, 102
- `\LARGE`, 58, 194, 209
- `\Large`, 45, 58, 194
- `\large`, 58, 191, 194
- \LaTeX , 2, 11
 - 调用程序, 13
 - 记号, 15
 - $\LaTeX 2_{\epsilon}$, 3
 - 与 2.09 版本的比较, 3–5
 - 与 2.09 版本的兼容性, 参见 兼容性
 - 模式, 12
 - 文档, 11
 - 文件, 11
 - 字体, 57–62
- `\LaTeX`, 15, 267
- $\LaTeX 3$ 项目, 3
- `\LaTeXe`, 16, 307
- \LaTeX 的编辑, 7–10
 - Edit, 7
 - EditPlus, 7
 - WinEdt, 9
- \LaTeX 计数器
 - chapter, 45, 159
 - enumn, 66, 159
 - equation, 159
 - errorcontextlines, 298
 - figure, 159
 - footnote, 87, 159
 - mpfootnote, 89, 159
 - page, 39, 159, 160, 166, 168
 - paragraph, 45, 159
 - part, 45
 - secnumdepth, 44, 46
 - section, 45, 159
 - subparagraph, 45, 159
 - subsection, 45, 159
 - subsubsection, 45, 159
 - table, 159
 - tocdepth, 46
- `latexsym.sty`(宏包), 204, 240
- `layout.sty`(宏包), 206
- `\lceil`, 102
- `\ldots`, 97

-
- `\le`, 240
 - `\leadsto`, 243
 - `\Leftarrow`, 243
 - `\leftarrow`, 243
 - `\leftarrowtail`, 243
 - `\lefteqn`, 110
 - `\leftharpoonup`, 243
 - `\leftleftarrows`, 243
 - `\leftmargin`, 70, 71–73, 161, 177, 179
 - `\leftmarginn`, 73
 - `\Leftrightarrow`, 243
 - `\leftrightarrow`, 243
 - `\leftrightarrows`, 243
 - `\leftrightsquigarrow`, 243
 - `\leftthreetimes`, 242
 - legalpaper选项, 34
 - `\lengthtest`, 168
 - `\leq`, 240
 - leqno选项, 34, 93, 101
 - `\leqslant`, 241
 - `\lessapprox`, 241
 - `\lessdot`, 241
 - `\lesseqgtr`, 241
 - `\lesseqqgtr`, 241
 - `\lessgtr`, 241
 - `\lesssim`, 241
 - `\let`, 276
 - letter类, 33, 204, 212
 - letter环境, 212
 - `\lfloor`, 102
 - lfonts.tex, 191
 - `\lg`, 244
 - `\lhd`, 242
 - `\lim`, 99, 244
 - `\liminf`, 99, 244
 - `\limits`, 99
 - `\limsup`, 99, 244
 - `\line`, 129, 305
 - `\linebreak`, 50, 52, 55, 150, 177, 321
 - `\linepenalty`, 53
 - `\linethickness`, 135
 - list环境, 68, 72, 300, 309
 - `\listfigurename`, 233
 - `\listfiles`, 181, 268, 273
 - `\listoffigures`, 47, 182, 315
 - `\listoftables`, 47, 182, 315
 - `\listparindent`, 70, 73
 - `\listtablename`, 233
 - `\ll`, 240
 - `\Lleftarrow`, 243
 - `\lll`, 241
 - `\ln`, 244
 - `\lnapprox`, 241
 - `\lneq`, 241
 - `\lneqq`, 241
 - `\lnsim`, 241
 - `\load`, 315
 - `\LoadClass`, 268, 270, 310, 311
 - `\location`, 212
 - `\log`, 244
 - `\Longleftarrow`, 243
 - `\longleftarrow`, 243
 - `\Longleftrightarrow`, 243
 - `\longleftrightarrow`, 243
 - `\longmapsto`, 243
 - `\Longrightarrow`, 243
 - `\longrightarrow`, 243
 - longtable.sty(宏包), 206
 - `\looparrowleft`, 243
 - `\looparrowright`, 243
 - `\lozenge`, 244
 - lplain格式, 56, 184, 191, 208
 - lplain.fmt, 56, 184
 - LR 盒子, 77–79
 - lrbox环境, 79, 307
 - `\ltimes`, 242
 - `\lvertneqq`, 241
 - `\magstep`, 337
 - `\magstephalf`, 337
 - `\mainmatter`, 45, 182
 - `\makeatletter`, 38, 101, 134, 169, 198, 233
 - `\makeatother`, 38, 101, 134, 169, 198, 233

-
- `\makebox`, 5, 77, 78, 79, 82
 - `\makebox(图画)`, 127, 128, 133, 137
 - `\makeglossary`, 189
 - `makeidx.sty`(宏包), 189
 - `MakeIndex`, 189–191, 204
 - 选项, 190
 - 样式文件, 191
 - `\makeindex`, 188
 - `\makelabel`, 69, 71, 177
 - `\makelabels`, 21, 213
 - `\MakeShortVerb`, 205
 - `\maketitle`, 43, 308, 318
 - `\mapsto`, 243
 - `\marginpar`, 90, 177, 307–309, 318
 - `\marginparpush`, 91
 - `\marginparsep`, 91
 - `\marginparwidth`, 91, 188
 - `\markboth`, 21, 37, 232
 - `\markright`, 37
 - `math`环境, 92
 - `\mathalpha`, 198
 - `\mathbf`, 4, 103, 111, 196, 306, 311
 - `\mathbin`, 198
 - `\mathcal`, 4, 97, 103, 196
 - `\mathclose`, 198
 - `\mathindent`, 35, 93, 110, 121
 - `\mathit`, 103, 196, 306
 - `\mathnormal`, 97, 103, 196
 - `\mathop`, 99, 198
 - `\mathopen`, 198
 - `\mathord`, 198
 - `\mathpunct`, 198
 - `\mathrel`, 198
 - `\mathring`, 100
 - `\mathrm`, 4, 99, 103, 112, 196
 - `\mathsf`, 103, 196, 315
 - `\mathtt`, 103, 196
 - `\mathversion`, 196, 320
 - `\max`, 99, 244
 - `\mbox`, 13, 77, 78, 79, 81, 102, 103, 106, 107, 111, 157, 266, 315
 - `\mddefault`, 195
 - `\mdseries`, 59, 195
 - `\measuredangle`, 244
 - `\medskip`, 51
 - `\medskipamount`, 51
 - `\MessageBreak`, 271, 272
 - `METAFONT`, 2, 207, 335
 - `\mho`, 244
 - Microsoft Word, 7, 11
 - `\mid`, 240
 - `\min`, 99, 244
 - `minipage`环境, 5, 79, 80, 83, 87, 110, 122, 127, 129, 307
 - minus, 18, 51, 71
 - mm(毫米), 17
 - `\models`, 240
 - `\month`, 20
 - `\mp`, 242
 - `mpfootnote`计数器, 89, 159
 - `\mu`, 240
 - `multicol.sty`(宏包), 206
 - `\multicolumn`, 104, 143, 147 308
 - `\multimap`, 243
 - `\multipt`, 126, 128, 130, 135, 137, 140, 142, 316
 - `\multlinegap`, 113
 - `multline`环境, 112
 - `multline*`环境, 113
 - `myheadings`页面样式, 37, 278
 - `\name`, 212, 213
 - `\natural`, 244
 - `\ncong`, 241
 - `\ne`, 241
 - `\nearrow`, 243
 - `\NeedsTeXFormat`, 6, 201, 265, 267, 268, 277, 310, 320
 - `\neq`, 241, 244
 - `\newboolean`, 169, 277, 278
 - `\newcommand`, 5, 120, 123, 162, 163–170, 173–177, 183, 190, 195, 201, 266, 270, 274–276, 298, 305–307, 314, 316

-
- `\newcommand*`, 271
 - `\newcounter`, 17, 69, 71, 159, 166, 170, 171, 174, 179, 182, 306, 308
 - `\newenvironment`, 5, 72, 170, 171–174, 266, 306, 316
 - `\newenvironment*`, 271
 - `\newfont`, 61, 196, 306
 - `\newif`, 277
 - `\newlength`, 17, 162, 165, 167, 169, 179, 278, 306
 - `newfont.sty`(宏包), 61, 204
 - `\newline`, 50, 150, 309, 321
 - `\newpage`, 46, 52, 77, 316
 - 禁止, 77
 - `\newsavebox`, 17, 78, 137, 172–174, 306, 314
 - `\newtheorem`, 73, 74, 159, 185, 306, 308, 316
 - `\nexists`, 244
 - NFSS, 3, 4, 6, 59, 61, 191–201, 264
 - 编码命令, 201
 - DC 字体, 192
 - 字体属性, 4, 58–61, 192–201, 326–335
 - 默认值, 195
 - `nfssfont.tex`, 203
 - `\ngeq`, 241
 - `\ngeqq`, 241
 - `\ngeqslant`, 241
 - `\ngtr`, 241
 - `\ni`, 240
 - `\nLeftarrow`, 243
 - `\nleftarrow`, 243
 - `\nLeftrightarrow`, 243
 - `\nleftrightarrow`, 243
 - `\nleq`, 241
 - `\nleqq`, 241
 - `\nleqslant`, 241
 - `\nless`, 241
 - `\nmid`, 241
 - `\nocorr`, 48, 60, 323
 - `\noexpand`, 276
 - `\nofiles`, 306
 - `\noindent`, 51, 171, 172
 - `\nolimits`, 99
 - `\nolinebreak`, 50, 52, 177
 - `\nonfrenchspacing`, 48
 - `\nonumber`, 108, 110, 112, 113
 - `\nopagebreak`, 52, 177
 - `\normalfont`, 60, 61, 195, 196
 - `\normalmarginpar`, 91
 - `\normalsize`, 58, 59, 194, 302, 312, 315
 - `\notag`, 112
 - `note`环境, 210
 - `\notin`, 241
 - `notitlepage`选项, 34
 - `\nparallel`, 241
 - `\nprec`, 241
 - `\npreceq`, 241
 - `\nrightarrow`, 243
 - `\nrightharpoonright`, 243
 - `\nshortmid`, 241
 - `\nshortparallel`, 241
 - `\nsim`, 241
 - `\nsubseteq`, 242
 - `\nsubseteqq`, 242
 - `\nsucc`, 241
 - `\nsucceq`, 241
 - `\nsupseteq`, 242
 - `\nsupseteqq`, 242
 - `\ntriangleleft`, 242
 - `\ntrianglelefteq`, 242
 - `\ntriangleright`, 242
 - `\ntrianglerighteq`, 242
 - `\nu`, 240
 - `null.tex`, 303
 - `\numberline`, 47
 - `\nVDash`, 242
 - `\nVdash`, 242
 - `\nvDash`, 242
 - `\nvdash`, 241
 - `\nwarrow`, 243
 - `\O`, 19, 239
 - `\o`, 19, 239
 - `\oddsidemargin`, 40, 41, 278

- \odot, 242
- \oint, 245
- oldfont.sty(宏包), 205
- \Omega, 240
- \omega, 240
- \ominus, 242
- onecolumn选项, 34
- \onecolumn, 41
- oneside选项, 34
- \onlynotes(slides), 211
- \onlyslides(slides), 211
- openany选项, 34
- openbib选项, 35
- \opening, 213
- openright选项, 34
- \oplus, 242
- \OptionNotUsed, 269
- \or, 277
- \oslash, 242
- OT1cmr.fd, 200
- OT1enc.def, 200
- \otimes, 242
- \oval, 131, 132, 135, 318
- \overbrace, 105
- Overfull \hbox警告, 321
- Overfull \vbox警告, 321
- overlay环境(slides), 210
- \overleftarrow, 106
- \overleftrightarrow, 106
- \overline, 105
- \overrightarrow, 106
- \P, 19, 243
- \PackageError, 266, 271
- \PackageInfo, 272
- \PackageWarning, 266, 272
- \PackageWarningNoLine, 272
- page计数器, 39, 159, 160, 166, 168, 182
- \pagebreak, 52, 53, 77, 177
- \pagenumbering, 17, 38, 39
- \pageref, 44, 158, 181, 185, 186, 318, 319
- \pagestyle, 37, 37, 184, 210
- \paperheight, 41
- \paperwidth, 41, 169
- \par, 51, 309, 315
- paragraph计数器, 45, 159
- \paragraph, 43
- \paragraph*, 43
- \parallel, 240
- \parbox, 5, 79, 80, 81, 84, 127, 129, 143, 150, 155, 266, 307, 308
- \parindent, 17, 39, 51, 73
- \parsep, 69, 71, 73, 177-179
- \parskip, 18, 39, 51, 69, 73, 161
- part计数器, 45
- \part, 43
- \part*, 43
- \partial, 243
- \partname, 233
- \PassOptionsToClass, 269
- \PassOptionsToPackage, 269
- patchdvi.exe, 226
- pc(pica), 17
- \penalty, 53
- \perp, 240
- \Phi, 240
- \phi, 240
- \Pi, 240
- \pi, 240
- pict2e.sty(宏包), 130, 131, 135, 141, 205
- picture环境, 85, 126, 205, 316
- picture环境的存贮, 139
- picture环境的广义语法, 139
- \pitchfork, 241
- plain.fmt, 56, 184
- Plain TeX, 2, 184
- 非法命令, 184
- plain页面样式, 37, 210
- plus, 18, 51, 71, 72
- \pm, 242
- \pmod, 99
- \poptabs, 76, 308
- \popziti, 232

-
- PostScript, 59, 191, 193, 194, 214
 - \pounds, 19, 239
 - \Pr, 244
 - \pr, 99
 - \prec, 240
 - \precapprox, 241
 - \preccurlyeq, 241
 - \preceq, 240
 - \precnapprox, 241
 - \precneqq, 241
 - \precsim, 241
 - \prime, 243
 - \printindex, 182, 189
 - proc类, 204
 - \ProcessOptions, 269, 273
 - \ProcessOptions*, 269, 278
 - \prod, 245
 - \projlim, 244
 - \propto, 240
 - \protect, 20, 46, 160, 183, 267, 271, 314, 317
 - \providecommand, 163, 201, 266, 270, 275
 - \providecommand*, 271
 - \ProvidesClass, 268, 306, 319, 320
 - \ProvidesFile, 268, 273
 - \ProvidesPackage, 268, 306, 319, 320
 - \ProvidesTextCommand, 201
 - \ProvideTextCommandDefault, 201
 - \ps, 213
 - \psi, 240
 - pt(点), 17
 - \pushtabs, 308
 - \pushziti, 232
 - \put, 126, 127-151
 - \qbezier, 134
 - \qqquad, 49, 114
 - \quad, 49, 114
 - quotation环境, 62, 63
 - quote环境, 16, 62, 63, 72
 - r.tex, 303
 - \r(“重音”), 19, 239
 - \raggedbottom, 41, 53
 - \raggedleft, 62
 - \raggedright, 62
 - \raisebox, 79, 83
 - \rangle, 102
 - \rceil, 102
 - \Re, 243
 - \ref, 44, 101, 108, 158, 181, 185, 318, 319
 - \refstepcounter, 159, 160
 - \relax, 276
 - \renewcommand, 5, 58, 66, 69, 88, 89, 101, 144, 154, 155, 162, 163, 165, 166, 174, 177, 200, 266, 270, 305, 314
 - \renewcommand*, 271
 - \renewenvironment, 5, 170, 172, 173, 266, 305, 314
 - \renewenvironment*, 271
 - report类, 33, 34, 37, 43-46, 68, 87, 155, 161, 166, 187, 204, 264, 302
 - \RequirePackage, 268, 269, 270, 277
 - \reversemarginpar, 91
 - \rfloor, 102
 - \rhd, 242
 - \rho, 240
 - \Rightarrow, 243
 - \rightarrow, 243
 - \rightarrowtail, 243
 - \rightharpoonowdown, 243
 - \rightleftarrows, 243
 - \rightleftharpoons, 243
 - \rightmargin, 70, 71, 72, 161
 - \rightrightarrows, 243
 - \rightsquigarrow, 243
 - \rightthreetimes, 242
 - \risingdotseq, 241
 - \rm, 4, 60, 112, 191
 - \rmdefault, 195, 200
 - \rmfamily, 59, 194, 200
 - Roman, 39
 - roman, 39
 - \Roman, 66, 88, 160
 - \roman, 88, 160

-
- `\Rrightarrow`, 243
 - `\rtimes`, 242
 - `\rule`, 15, 82, 90, 149, 151, 155, 176
 - `\S`, 239, 244
 - `s.tex`, 303
 - `\samepage`, 53
 - `sample.tex`, 203
 - `sample2e.tex`, 203
 - `\savebox`, 5, 78, 84
 - `\savebox(图形)`, 137, 138, 139, 142
 - `\sbox`, 78, 79, 84, 150, 173, 266
 - `\sc`, 60, 191
 - `\scdefault`, 195
 - `\scriptfont`, 315
 - `\scriptscriptfont`, 315
 - `\scriptscriptstyle`, 118, 198
 - `\scriptsize`, 58, 194
 - `\scriptstyle`, 118, 198
 - `\scshape`, 59, 195
 - `\searrow`, 243
 - `\sec`, 244
 - `secnumdepth`计数器, 44
 - `section`计数器, 45, 159, 161
 - `\section`, 38, 43, 87
 - `\section*`, 43
 - `\see`, 189, 190
 - `\seename`, 190
 - `\selectfont`, 167, 193, 194, 195, 198
 - `\seriesdefault`, 195
 - `\setboolean`, 169, 269, 277, 278
 - `\setbox`, 266
 - `\setcounter`, 17, 39, 44–46, 87, 101, 154, 160, 166, 170, 174, 182, 298, 308, 316
 - `\setlength`, 16–18, 35, 39, 41, 51, 58, 69, 72, 73, 76, 85, 89, 91, 113, 161, 169, 177–179
 - `\SetMathAlphabet`, 197, 312
 - `\setminus`, 242
 - `\SetSymbolFont`, 197
 - `\settime(slides)`, 211
 - `\settodepth`, 5, 161
 - `\settoheight`, 5, 161
 - `\settowidth`, 5, 161, 165, 177, 179
 - `\sf`, 60, 191, 315
 - `\sf@size`, 198
 - `\sfdefault`, 195, 200
 - `\sffamily`, 59, 61, 194, 200, 209
 - `\shapedefault`, 195
 - `\sharp`, 244
 - `\shortmid`, 241
 - `\shortparallel`, 241
 - `\shortstack`, 132
 - `shortvrb.sty(宏包)`, 85, 205
 - `\shoveleft`, 113
 - `\shoveright`, 113
 - `\showhyphens`, 56
 - `showidx.sty(宏包)`, 188, 205
 - `showkeys.sty(宏包)`, 206
 - `\Sigma`, 240
 - `\sigma`, 240
 - `\signature`, 212, 213
 - `\sim`, 240
 - `\simeq`, 240
 - `\sin`, 244
 - `\sinh`, 244
 - `\sl`, 60, 191
 - `\sldefault`, 195
 - `slide`环境 (slides), 209
 - `slides`类, 204, 208–211
 - SL_TEX, 参见 幻灯片
 - `\sloppy`, 55, 80, 321, 322
 - `sloppypar`环境, 55, 321, 322
 - `\slshape`, 45, 48, 59, 60, 71, 72, 171–173, 179, 195, 294, 295, 300
 - `\small`, 16, 58, 59, 194
 - `small.tex`, 203
 - `small2e.tex`, 203
 - `\smallfrown`, 241
 - `\smallint`, 243
 - `\smallsetminus`, 242
 - `\smallskip`, 51
 - `\smallskipamount`, 51

-
- \smallsmile, 241
 - \smile, 240
 - somedefs.sty(宏包), 206
 - \spadesuit, 244
 - \sphericalangle, 244
 - splain格式, 208
 - \sqcap, 242
 - \sqcup, 242
 - \sqrt, 95
 - \sqsubs, 240
 - \sqsubseteq, 240
 - \sqsupset, 240
 - \square, 244
 - \SS, 19, 239
 - \ss, 19, 239
 - \ssf@size, 198
 - \stackrel, 106
 - \star, 242
 - \stepcounter, 85, 89, 159, 160, 166, 171–173
 - \stretch, 162
 - \subarray, 107
 - \subitem, 187, 188, 189
 - subparagraph计数器, 45, 159
 - \subparagraph, 43
 - \subparagraph*, 43
 - \Subset, 241
 - subsection计数器, 45, 159
 - \subsection, 43
 - \subset, 240
 - \subseteq, 240
 - \subseteqq, 241
 - \subsetneq, 242
 - \subsetneqq, 242
 - \substack, 107
 - \subsubitem, 187, 188, 189
 - subsubsection计数器, 45, 159
 - \subsubsection, 43
 - \subsubsection*, 43
 - \succ, 240
 - \succapprox, 241
 - \succcurlyeq, 241
 - \succeq, 240
 - \succnapprox, 241
 - \succnsim, 241
 - \succsim, 241
 - \sum, 245
 - \sup, 99, 244
 - \suppressfloats, 5, 153, 156
 - \Supset, 241
 - \supset, 240
 - \supseteq, 240
 - \supseteqq, 241
 - \supsetneq, 242
 - \supsetneqq, 242
 - \surd, 243
 - \swarrow, 243
 - \symbol, 61, 202
 - syntonly.sty宏包, 205
 - \t(重音), 19, 239
 - tabbing环境, 74, 308, 309
 - \tabbingsep, 76
 - table计数器, 159
 - table环境, 142, 152, 155, 156, 158, 184, 185, 307, 318
 - table*环境, 152
 - \tablename, 233
 - \tableofcontents, 46, 47, 182, 315, 323
 - tabular环境, 142, 144, 145, 307, 308, 313, 314, 316, 317
 - tabular*环境, 142, 144
 - \tabularnewline, 144
 - tabularx.sty(宏包), 206
 - \tag, 112
 - \tan, 244
 - \tanh, 244
 - \tau, 240
 - \telephone, 212
 - testpage.tex, 203
 - \testquotelleft, 203
 - \TeX, 16
 - TeXGuru, 8, 10, 208
 - \textbf, 60, 179, 195, 196

-
- \textbullet, 203
 - \textcircled, 202
 - \textcompwordmark, 202
 - \textemdash, 203
 - \textendash, 203
 - \textexclamdown, 203
 - \textfloatsep, 154
 - \textfont, 315
 - \textfraction, 154
 - \textheight, 40, 41, 53, 321
 - \textit, 48, 60, 195
 - \textmd, 60, 195
 - \textnormal, 60, 195
 - \textperiodcentered, 203
 - \textquestiondown, 203
 - \textquotedblleft, 203
 - \textquotedblright, 203
 - \textquoteright, 203
 - \textrm, 60, 195
 - \textsc, 60, 195
 - \textsf, 60, 195
 - \textsl, 48, 60, 195
 - \textstyle, 106, 118, 120
 - \textsuperscript, 202
 - \texttt, 60, 177, 195
 - \textup, 60, 195
 - \textvisiblespace, 203
 - \textwidth, 18, 40, 41, 161, 169, 271, 277, 279, 298
 - \tf@size, 198
 - \tfrac, 94
 - \thanks, 21, 42, 313
 - \the计数器, 161
 - thebibliography环境, 67, 68, 186, 309
 - \theequation, 166
 - \thefootnote, 88, 166
 - theindex环境, 187, 188-190
 - theorem环境, 74
 - theorem.sty(宏包), 206
 - \thepage, 161
 - \therefore, 241
 - \thesection, 161
 - \Theta, 240
 - \theta, 240
 - \thickapprox, 241
 - \thicklines, 129, 130, 135, 136, 138, 139
 - \thicksim, 241
 - \thinlines, 126, 135, 136
 - \thispagestyle, 17, 37, 210
 - \tilde, 244
 - \times, 242
 - times.sty, 200
 - \tiny, 58, 191, 194
 - \title, 42, 43, 308, 318
 - titlepage环境, 42, 43
 - titlepage选项, 34, 43
 - \to, 243
 - tocdepth计数器, 46
 - \today, 20
 - \top, 243
 - \topfigrule, 154
 - \topfraction, 154
 - \topmargin, 40, 278
 - topnumber, 154
 - \topsep, 122, 179
 - \topskip, 40, 41
 - \totalheight, 78
 - totalnumber, 154
 - tracefnt.sty(宏包), 205, 320
 - \triangle, 243
 - \triangledown, 244
 - \triangleleft, 242
 - \trianglelefteq, 241
 - \triangleq, 241
 - \triangleright, 242
 - \trianglerighteq, 241
 - trivlist环境, 72
 - \tt, 60, 62, 191
 - \ttdefault, 195, 200
 - \ttfamily, 59, 177, 194, 200
 - twocolumn选项, 34, 35, 36, 41
 - \twocolumn, 41, 52

-
- `\twoheadleftarrow`, 243
 - `\twoheadrightarrow`, 243
 - `twoside`选项, 34, 35, 37, 41, 52, 53
 - `\typein`, 21, 183, 211
 - `\typeout`, 21, 182, 183
 - `\u`(\sim 重音), 19, 239
 - `\unboldmath`, 111, 196
 - `\underbrace`, 105
 - `Underfull \hbox`警告, 321
 - `Underfull \vbox`警告, 322
 - `\underleftarrow`, 106
 - `\underlefterightarrow`, 106
 - `\underline`, 105
 - `\underrightarrow`, 106
 - `\unitlength`, 125, 126, 128, 135, 136
 - `\unlhd`, 242
 - `\unrhd`, 242
 - `\Uparrow`, 102, 243
 - `\uparrow`, 102, 243
 - `\updefault`, 195
 - `\Updownarrow`, 102, 243
 - `\updownarrow`, 102, 243
 - `\upharpoonleft`, 243
 - `\upharpoonright`, 243
 - `\uplus`, 242
 - `\upshape`, 59, 195
 - `\Upsilon`, 240
 - `\upsilon`, 240
 - `\upuparrows`, 243
 - `\usebox`, 79, 84, 137, 138, 150, 151
 - `\usecounter`, 69, 71, 72, 179
 - `\usefont`, 193
 - `\usepackage`, 4, 35, 36, 98, 167, 188, 189, 200, 202, 203, 205, 209, 214, 228, 264, 268, 273, 274, 279, 303, 306, 307, 310, 311, 319, 320
 - `\v`(\sim 重音), 19, 239
 - `\value`, 21, 160, 166, 168, 170, 314
 - `\varDelta`, 240
 - `\varepsilon`, 240
 - `\varGamma`, 240
 - `\varinjlim`, 244
 - `varioref.sty`(宏包), 206
 - `\varkappa`, 240
 - `\varLambda`, 240
 - `\varliminf`, 244
 - `\varlimsup`, 244
 - `\varnothing`, 244
 - `\varOmega`, 240
 - `\varPhi`, 240
 - `\varphi`, 240
 - `\varPi`, 240
 - `\varpi`, 240
 - `\varprojlim`, 244
 - `\varpropto`, 241
 - `\varrho`, 240
 - `\varSigma`, 240
 - `\varsigma`, 240
 - `\varsubsetneq`, 242
 - `\varsubsetneqq`, 242
 - `\varsupsetneq`, 242
 - `\varsupsetneqq`, 242
 - `\varTheta`, 240
 - `\vartheta`, 240
 - `\vartriangle`, 244
 - `\vartriangleleft`, 241
 - `\vartriangleright`, 241
 - `\varUpsilon`, 240
 - `\varXi`, 240
 - `\vbox`, 266
 - `\Vdash`, 241
 - `\VDash`, 241
 - `\vdash`, 240
 - `\vdots`, 97
 - `\vec`, 244
 - `\vector`, 130, 131, 133, 135, 136, 138, 139
 - `\vee`, 242
 - `\veebar`, 242
 - `\Vert`, 243
 - `\verb`, 85, 204, 310
 - `\verb*`, 85
 - `verbatim`环境, 85

- verbatim*环境, 85
 - verbatim.sty(宏包), 206
 - verse环境, 63
 - \vfill, 51
 - \visible(slides), 210
 - \vline, 144
 - \vspace, 51, 53, 83, 90, 91, 177, 309
 - \vspace*, 51, 154
 - \Vdash, 241
 - \wedge, 242
 - \whiledo, 167, 169
 - \widehat, 244
 - \widetilde, 244
 - \widowpenalty, 53
 - \width, 78, 82
 - WinEdt, 9
 - \wp, 243
 - WPS, 11
 - \wr, 242
 - x.tex, 303
 - \xdef, 266, 276
 - \Xi, 240
 - \xi, 240
 - \xleftarrow, 106
 - xr.sty(宏包), 206
 - \xrightarrow, 106
 - xspace.sty(宏包), 206
 - \year, 20
 - \zeta, 240
 - \zihao, 229
 - \ziju, 231
 - \ziti, 230
-
- 版权符号, 19, 239
 - 包装语言, 1, 3
 - 编辑程序, 1
 - 变量, 参见 公式
 - 编码命令 (NFSS), 201
 - 编码 (字体属性), 192
 - 边注, 90-91
 - 定位
 - 标准, 90
 - \twocolumn, 90
 - \twoside, 90
 - 相反, 90
 - 样式参数, 91
 - 标尺盒子, 82
 - 表格, 142-151
 - 编号, 参见 浮动对象
 - 表格中的脚注, 88
 - 标题, 参见 浮动对象
 - 浮动, 参见 浮动对象
 - 构造, 142-144
 - @- 表达式, 142-144, 147
 - 段落列, 143, 149
 - 行结束, 143
 - 列重复, 104, 143
 - 列分隔, 143
 - 列格式, 142
 - 列合并, 143, 147
 - 水平线, 143
 - 竖线, 143, 144
 - 数学中的域, 103-105
 - 竖直定位参数, 142
 - 宽度, 142
 - 列对齐, 146
 - 列间距, 143
 - 示例, 参见 表格样例
 - 数学表格, 103-105
 - 样式参数, 144
 - 表格样例, 144-151
 - 标题
 - 表格, 参见 浮动对象
 - 章节, 43
 - 标题页, 34, 42
 - 标准格式, 42
 - 无格式, 43
 - 不可见命令, 176
 - 彩色层, 参见 投影片
 - 彩色投影片, 参见 投影片
 - 参考文献, 64, 180, 186-188

- BibTeX, 187, 188
- openbib, 35
- 条目, 67, 186
- 在正文中的引用, 66, 186
- 参数, 15
 - 不可省的, 15
 - 脆弱的参数, 20
 - 可省的, 15
 - 移动参数, 20
- 长度, 17, 161
 - 单位, 17
 - 赋值, 参见 \setlength
 - 固定长度, 17
 - 零, 17
 - 零长度, 17
 - 弹性长度, 参见 弹性长度
- 长度单位, 17
- 插入外部图形, 214
- 插图, 参见 浮动对象
- 插图列表, 47
- 常量, 参见 公式
- 程序设计, 264–277
 - 错误消息, 271
 - 警告, 272
 - 兼容模式, 274
 - 牢固的命令, 270
 - 例子, 277
 - 上载文件, 268
 - 稳定性, 267
 - 文件检测, 267–268
 - 选项, 269–270
 - 延迟代码, 270
 - 指导方针, 266
- 重叠文本, 78
- 脆弱命令, 20, 267
- 错误消息
 - 出错行, 292, 295
 - 错误标识, 292
 - 错误的传播, 299–301
 - 多文件错误, 304–305
 - 紧急停止, 303
 - 来自 L^AT_EX 的错误, 294–297
 - 来自 L^AT_EX 2.09 的错误, 297–298
 - 来自 T_EX 的错误, 292–294
 - 来自 T_EX 宏的错误, 298
- 清单
 - L^AT_EX 宏包错误, 310–311
 - L^AT_EX 宏包警告, 319–320
 - L^AT_EX 普通错误消息, 305–310
 - L^AT_EX 普通警告, 317–319
 - L^AT_EX 字体错误, 311–313
 - L^AT_EX 字体警告, 320–321
 - 难以找到的错误, 322
 - T_EX 错误, 313–317
 - T_EX 警告, 321–322
- 数学错误, 303–304
- T_EX 中错误消息的基本结构, 294
- 未知文件名, 302, 307
- 用户反应
 - 帮助, 293
 - 调用编辑器, 293
 - 校正, 293
 - 继续处理, 293
 - 删除, 293
 - 推荐方法, 297
- 中止程序
 - 利用编辑器, 293
 - 利用 I\stop, 293, 301, 309
 - 利用 X, 293, 301
- 打印机驱动程序, 2, 14, 141, 184
 - 24 针打印机驱动程序, 14
 - HP 激光打印机驱动程序, 14
 - 北佳激光打印机驱动程序, 14
 - 喷墨打印机驱动程序, 14
- 单独公式, 92
- 单字符命令, 15
- 导言 (preamble), 11
- 底边, 41
- 顶边, 40
- 定理, 73
- 断词, 54–56, 316, 321

- 多语言文本的断词, 56
- 关闭断词, 55
- 过满 `\hbox`, 321
- 例外列表, 54, 55
- 屏幕列表, 56
- 人工断词, 54
- 有重音字母的断词, 315
- 断行
 - 不鼓励, 50
 - 额外间距, 50
 - 鼓励, 50
 - 居中文本, 62
 - 强制, 50
 - 抑制, 48, 50
 - 右对齐, 62
 - 左对齐, 62
- 段落, 11
 - 盒子, 77, 79
- 段落间距, 参见 间距
- 段落结束
 - 用空行, 11
 - 用 `\par`, 51
- 多行公式, 108
- 多文件文本, 180
 - 合并, 180
 - 有选择地处理, 181
- 多语种 L^AT_EX, 207
- 多字符命令, 15
- 二项式系数, 106
- 二元运算符号, 242
- 方根, 95
- 方根符号, 95
- 分段
 - 额外行间距, 11, 18, 39, 51
 - 首行缩进, 11, 39
 - 取消, 51
 - 设置, 17, 39, 51
 - 章节命令后, 51
- 分列, 52
- 分数, 94
- 分页, 11, 51
 - 不鼓励, 52
 - 鼓励, 52
 - 两列模式中的, 52
 - 两面模式中的, 52
 - 强制, 52
 - 左右调整, 52
 - `\tabbing` 中的, 77
 - 有图表时的, 52
 - 有限制的, 53
- 否定符号, 98
- 浮动对象, 151–158
 - 安置, 152
 - 选择准则, 153
 - 延迟, 153
 - 抑制, 153
- 编号, 155
- 表格标题, 155
- 标题, 155
 - 宽度, 155
- 插图标题, 155
- 两列, 152
- 使缓冲区溢出, 316
- 示例, 156–158
- 样式参数, 154
- 负号, 19
- 符号
 - 堆积, 106
 - 二元运算符, 98, 242
 - 否定, 98, 243
 - 关系运算符, 98, 240
 - 键盘符号, 93
 - 箭头, 98, 243
 - 积分符号, 95
 - 括号, 101
 - 不可见的, 102
 - 多行公式中, 109, 121
 - 手工调整尺寸, 120
 - 自动调整尺寸, 101
 - 上下限, 96, 98
 - 替换命令, 202

- 有两种尺寸的符号, 245
- 杂类, 98, 243
- 指针, 98, 243
- 字母
 - 花体, 97
 - 希伯来字母, 240
 - 希腊字母, 97, 240
- 附录, 45
- 辅助文件, 186
- 格式化程序, 1
- 格式文件, 2, 208, 267
- 公式编号
 - 右对齐, 92, 101
 - 自定义, 166
 - 竖直居中, 101
 - 左对齐, 93, 101
- 公式中的黑体, 111
- 关键词索引, 参见 索引记录
- 关系运算符, 98, 240
- 固定长度, 17
- 国际化 L^AT_EX, 217
- 函数名称, 99, 244
- 行间距, 参见 间距
- 行宽, 41
- 行列式, 103–105
- 黑线, 参见 标尺
- 盒子, 77–85
 - 标尺盒子, 77, 82
 - 存贮盒子, 78
 - 调用盒子, 79
 - 定位
 - 水平定位, 77
 - 竖直定位, 79, 81, 82
 - 段落, 77
 - LR, 77, 78
 - 嵌套, 83
 - 上升, 79
 - 竖直盒子, 79
 - 下降, 79
 - 样式参数, 84
 - 有框盒子, 77
 - 子段盒子, 79
- 宏包, 3, 4, 35
 - 标准, 204
 - 附属宏包, 207
 - afterpage, 206
 - alltt, 86, 204
 - amsmath, 94, 97, 101, 106, 107, 240, 242
 - amssymb, 98, 240–244
 - array, 206
 - bezier, 134, 204
 - CJK, 218
 - color, 209, 266
 - dcolumn, 206
 - delarray, 206
 - doc, 204
 - enumerate, 206
 - epsfig, 215
 - eufrak, 98
 - exscale, 204
 - fileerr.dtx, 206
 - fontenc, 200, 204
 - fontsmpl, 206
 - ftnright, 206
 - fullpage, 277
 - graphics, 214
 - graphicx, 214
 - graphpap, 141, 204
 - hhline, 206
 - ifthen, 167, 204, 266, 273, 277
 - indentfirst, 206
 - inputenc, 204
 - latexsym, 204, 240, 242–244
 - layout, 206
 - longtable, 206
 - makeidx, 189, 204
 - multicol, 206
 - newlfont, 61, 204
 - oldlfont, 205
 - pict2e, 130, 131, 135, 141, 205
 - shortvrb, 85, 205

- showidx, 205
- showkeys, 206
- somedefs, 206
- syntonly, 205
- tabularx, 206
- theorem, 206
- Tienc, 205
- tracefmt, 205, 320
- varioref, 206
- verbatim, 206
- xx, 206
- xspace, 206
- 环境, 16
 - 没有名称的环境, 16, 57, 58
 - 命令名称做为环境名, 16
 - 数学环境, 92
- 幻灯片, 208–211
 - 彩色层, 208
 - 时间注解, 211
 - 首页, 209
 - 页面样式, 210
 - 有选择地处理, 211
 - 注解, 210
- 花体字母, 97
- 化学方程式, 111, 167
- 回车, 11, 18
- 汇总, 189
- 积分符号, 95
- 键盘符号, 93
- 键盘输入, 183
- 箭头, 98, 106, 243
- 交叉引用, 185, 186
 - 表格, 158, 185
 - 参考文献, 67, 186, 187
 - 插图, 158, 185
 - 定理, 185
 - 公式, 101, 108, 185
 - 计数器, 160, 186
 - 页码, 185
 - 章节, 185
- 交换图, 116
- 脚注, 87–90
 - 标准标记, 87
 - 标准脚注, 87
 - 禁止, 87
 - 表格中, 89
 - 非标准脚注, 87
 - 改变标记, 88
 - 禁止模式, 88
 - 小页环境中, 87, 89
 - 样式参数, 89
- 间距
 - 不期望的间距, 176
 - 单词间距, 47
 - 标点符号后的, 47
 - 句号后的, 47
 - 糟糕程度, 322
 - 段落间距, 11, 18, 39, 51
 - \frenchspacing, 48
 - 行间距, 11
 - 标准, 58
 - 段尾, 39
 - 改变, 39
 - 局部修改, 51
 - 竖直间距, 50, 51
 - 弹性长度, 51
 - 手工改变, 47
 - 水平间距, 48
 - 字符间距, 47
- 基线, 39, 41, 59
- 计数器, 参见 L^AT_EX 计数器
- 计数器的值, 160
 - 显示, 160
- 计算机现代字体, 4, 59, 191–194, 200, 324, 325
- 居中文本, 62
- 居中与缩进, 62–63
- 矩阵, 103–105
 - 正文中的小矩阵, 107
- 空白, 18
 - 多个, 11, 18

- 强迫的, 18
- 弹性, 49
- 消除空白, 18
 - 一行开头的, 18
- 空格, 11, 12, 15
 - 命令后的, 15
- 空行开始新段, 18
- 括号符号, 参见 符号
- 牢固的命令, 20, 270
- 连分数, 119, 122
- 连写, 20, 325
 - 断开, 20
- 连续点, 96
- 连字符, 19
- 两列页面格式
 - 列间分隔线, 35
- 两字符命令, 15
- 列表, 63–73
 - 编号, 69
 - 标准标签, 69
 - 格式示意图, 70
 - 平凡列表, 72
 - 嵌套, 64, 65, 73
 - 样式参数, 69
 - 用户环境, 72
- 列表标签, 63
 - 改变样式, 65
- 浏览程序, 13
 - cctwin, 13
 - view.bat, 13
- 美国拼写, 169
- 美国数学会, 207
- 美元符号, 19, 239
- 命令, 11, 15
 - *- 形式, 15
 - 不可见的, 176
 - 层次, 265
 - 脆弱的命令, 20
 - 单字符命令, 15
 - 多字母命令, 15
 - 后接空格, 15, 163
 - 牢固的命令, 20
 - 两字符命令, 15
 - 特殊的命令, 15
 - 语法, 15
 - 与普通文本区分, 11
- 命令字符的显示, 19, 239
- 模式
 - 从左到右模式, 12, 13, 77
 - 段落模式, 12, 13, 79
 - 数学模式, 12, 13, 92
- 目录表, 46–47
 - 表格的, 47
 - 插图的, 47
 - 手工加入的条目, 46
 - 条目层次, 46
 - tocdepth, 46
 - 显示, 46
 - 自动条目, 46
- 拼写方式 (美国或英国方式), 169
- 平凡列表, 72
- 破折号, 19
 - 短破折号, 19
 - 负号, 19
 - 连字符, 19
 - 全破折号, 19
- 倾斜校正, 57, 60
- 其他字体, 61
- 求和符号, 95
- 上标, 93
- 省略号, 96
- 声明, 16
 - 范围, 16
 - 局部性, 16
 - 全局性, 17
- 伸展值, 参见 弹性长度
- 诗歌, 63
- 输入编码, 202
- 书信, 211–214

- 称呼, 213
- 分发名单, 213
- 附件, 213
- 回信地址, 212
- 结尾, 213
- 收信人姓名与地址, 213
- 属性 (字体), 4, 58-61, 192-200, 204
 - 默认值, 195
- 数学变体, 197
- 数学公式, 92
 - 编号, 92
 - 变量, 93
 - 并列, 110
 - 常量, 93
 - 单独公式
 - 居中, 92
 - 左对齐, 92
 - 单独公式, 92
 - 多行公式, 93, 108-110
 - 编号, 108
 - 分行, 108
 - 括号尺寸, 109, 120
 - 列分隔符号, 108
 - 方根, 95
 - 分数, 94
 - 公式中的空白, 93
 - 函数名称, 99, 244
 - 黑体, 111
 - 化学方程式, 112, 167
 - 积分, 95
 - 精调, 117-122
 - 括号的尺寸, 120
 - 水平间距, 117
 - 字体尺寸, 118-120
 - 矩阵, 103-105
 - 连分数, 119, 122
 - 连续点, 96
 - 求和, 95
 - 上划线, 105
 - 上括号, 105
 - 上下限
 - 定位, 96, 99
 - 函数的上下限, 99
 - 具有两种尺寸, 98
 - 省略号, 96
 - 水平间距, 96, 117
 - 数学样式参数, 121
 - TeX 命令, 106
 - 下划线, 105
 - 下括号, 105
 - 希腊字母, 97, 240
 - 有框公式, 110, 122
 - 域, 103-105
 - 正文公式, 92
 - 指标, 93
 - 指数, 93
 - 字体尺寸, 118
 - 简化情形, 118
 - 详细情形, 120
 - 数学精调, 117-122
 - 括号尺寸, 120
 - 水平间距, 117
 - 字体尺寸, 118-120
 - 数学模式, 92
 - 数学样式参数, 121
 - 数学重音, 99
 - 数学字母, 103, 196
 - 双列页面格式, 34
 - 列间距, 35
 - 双面格式, 34
 - 所见即所得, 1, 7
 - 索引记录, 187-191
 - 弹性长度, 18, 39, 71, 77, 78, 82, 122, 154, 161, 162, 179, 266
 - 特殊字符, 18-20
 - 天元系统, 220
 - 条件文本, 167
 - 调整, 11, 47
 - 图形
 - 半圆, 131
 - 存贮, 137-139
 - 单位长度, 125

- 定位命令, 126
- 多行文本, 127
 - 参考点, 127
- 浮动图形, 参见 浮动对象
- 更多例子, 140
 - 重复命令, 140
 - 网格, 140
- 盒子, 127-129
 - 定位参数, 128
 - 示例, 128
 - 实线边框, 127, 128
 - 无边框, 127
 - 虚线边框, 127, 129
- 画图命令, 127-151
- 箭头, 130
 - 可取的斜率, 130
 - 最小长度, 130
- 偏移, 139
- 卵形线, 131
 - 示例, 131
- 嵌套, 135, 137
 - 新单位长度, 136
- 曲线, 134
- 竖直堆积文本, 132
- 四分之一圆周, 131
- 图形元素, 126
 - 半圆, 131
 - 重复命令, 126
 - 定位, 126
 - 盒子, 127-129
 - 箭头, 130
 - 卵形线, 131
 - 四分之一圆周, 131
 - 文本, 127
 - 圆, 130
 - 直线, 129
- x 轴, 125
- 一般性建议, 141
- 有框文本, 133
- 圆, 130
 - 空心圆, 130
 - 实心圆, 130
- y 轴, 125
- 直线, 129
 - 可使用的斜率, 129
 - 斜线, 129
 - 最短长度, 130
 - 直线粗细, 135
 - 坐标系, 125
- 网络服务器, 207
- 文本
 - 重叠, 78
 - 多文件, 180
 - 公式中的文本, 102
 - 合并, 180
 - 宽度计算, 161
 - 竖直堆积, 132
 - 提升, 79
 - 下划线, 105
 - 下降, 79
 - 显示源文本, 85
 - 有框文本, 77
 - 有选择地处理, 181
 - 正文公式, 92
 - 正文中的命令, 11
 - 正文中的注释, 86
 - 组成, 11
- 文档的主要组成, 42-45
- 文档类, 4, 33
 - 选项, 33
- 文件
 - 抄本, 181, 199, 205, 271, 272
 - 输入文件, 273
 - 文件清单, 273
 - 稳妥输入文件, 272
- 无名环境, 16, 57
- 下标, 93
- 显示文本
 - 单边调整, 62
 - 居中, 62
 - 缩进, 62

- 小页, 80, 81, 84, 89
 - 在 L^AT_EX 2_ε 中的小页, 81
- 显示源文本, 85, 86
- 新字体选择框架, 参见 NFSS
- 选项
 - 处理, 269
 - 定义, 269
 - 宏包选项, 36
 - 局部的, 36, 270
 - 类选项, 33
 - 全局的, 36, 270
- 样式 (L^AT_EX 2.09), 3, 264
- 样式参数
 - 边注, 91
 - 表格, 144
 - 浮动, 154
 - 盒子, 84
 - 脚注, 89
 - 列表, 69
 - 数学, 121
- 样式选项, 36
- 页边界, 40
- 页边记号, 90
- 脚码, 40
 - 阿拉伯数字样式, 39
 - 标题页, 37
 - 大写罗马字母样式, 39
 - 大写字母样式, 39
 - 改变, 39
 - 去掉, 37
 - 设置, 39
 - 小写罗马字母样式, 39
 - 小写字母样式, 39
- 书眉, 40
- 页面格式, 34, 40
 - 单列, 34, 41
 - 单面, 34
 - 满页, 277–279
 - 设置, 41
 - 示意图, 40
 - 双列, 34, 41
 - 双面, 34
- 页面样式
 - empty, 37, 210, 278
 - headings, 37, 43, 210, 212, 213
 - myheadings, 37, 278
 - plain, 37, 210, 278
- 英国拼写, 169
- 引号, 18, 48
- 用户定义的命令, 162–167
 - 保存, 173
 - 不期望的空白, 176
 - 后接空格, 163
 - 例子
 - 给文本加框, 165
 - 公式编号, 166
 - 化学方程式, 167
 - 使 T_EX 命令具有 L^AT_EX 语法, 165
 - 自定义脚注, 166
 - 没有参数的自定义命令, 162, 163
 - 调用, 162
 - 嵌套, 175
 - 顺序, 174
 - 缩写结构, 174
 - 同时适用于数学与文本模式, 163
 - 一般性建议, 173
 - 有参数的自定义命令, 163, 164
 - 传递参数, 175
 - 调用, 164
 - 有一个可省参数的自定义命令, 165
 - 作用范围, 174
 - 局部性, 174
 - 全局性, 174
- 用户自定义的计数器, 159
 - 创建, 159
 - 递增, 160
 - 改变, 160
 - 设置, 160
 - 自动重设, 159
- 用户自定义长度
 - 创建, 162

- 放大, 162
- 设置, 161
- 文本长度的计算, 161
- 增加, 161
- 用户自定义的环境, 170
 - 保存, 173
 - 没有参数的环境, 170
 - 调用, 171
 - 示例
 - genlist, 178
 - ttscript, 177
 - 一般性建议, 173
 - 有参数的环境, 172, 173
 - 调用, 172
 - 有一个可省参数的环境, 173
 - 作用范围, 174
 - 局部性, 174
 - 全局性, 174
- 有框表格, 145
- 有框公式, 110, 122
- 有框文本, 77
- 有选择地处理文件, 181
- 域, 103-105
- 与 2.09 版本的兼容性, 4, 5
 - @compatibility 布尔变量, 275
 - \@options, 264
 - article.sty, 318
 - \bezier, 134
 - bezier.sty, 134, 204
 - \bf, 60
 - 不能用 L^AT_EX 2_ε 命令, 307
 - \documentstyle, 12
 - \footheight, 41
 - \it, 60
 - oldfont, 205
 - proc.sty, 204
 - \rm, 60
 - \sc, 60
 - \sf, 60
 - \sl, 60
 - \tt, 60
 - 字体尺寸命令, 58
 - 字体样式命令, 60
 - 圆弧, 123, 134
 - 在文件中包含另一个文件, 274
 - 摘要, 42
 - 糟糕程度 (单词间距), 321
 - 章节
 - *- 形式, 43
 - 编号, 43
 - 层次, 44
 - 取消, 43
 - secnumdepth, 44
 - 标题, 43
 - 缩短形式, 43
 - 序列, 43
 - 章节命令, 43
 - 正文公式, 92
 - 正文 (body), 11, 40
 - 正文高度, 41
 - 制表, 74-77
 - 存贮, 76
 - 其他命令, 76
 - 其中的重音, 76
 - 设置制表位, 74
 - 向后, 75
 - 样本行, 75
 - 左边界, 75
 - 制表位, 74
 - 支撑, 83, 149, 150
 - 指数, 93
 - 纸张大小选项, 34
 - 重音
 - 数学的, 99
 - 在其他语言中, 19
 - 在 tabbing 环境中, 76
 - 注释, 86
 - 注释符号, 86
 - 自动补足, 7
 - 子段盒子, 79-81, 83, 87
 - L^AT_EX 2_ε 中的子段盒子, 81

- 字符指定, 61
 - 八进制, 61
 - 十进制, 61
 - 十六进制, 61
- 字母
 - 花体字母, 97
 - 希伯来字母, 240
 - 希腊字母, 97, 240
- 字体, 324
 - 成比例字体, 326–331
 - cminch 字体, 330
 - Cyrillic, 191
 - 打字机字体, 参见 固定字体
 - DC, 192
 - 分类, 324
 - 固定字体, 324, 331
 - 黑体, 328
 - 记号字体, 336
 - 计算机现代字体, 4, 59, 191–193, 200, 324, 325
 - 罗马字体, 326
 - 名称, 336
 - 命令, 60, 194
 - NFSS, 191–201
 - NFSS 的定义, 198–200
 - PostScript 字体, 191, 193, 196, 200, 207
 - 其他, 61
 - Sans serif 黑体, 330
 - 属性, 59, 60, 192–201
 - 编码, 192
 - 尺寸, 59, 192
 - 默认值, 195
 - 内部命令, 198
 - 形状, 59, 192
 - 序列, 59, 192
 - 族, 59, 192
 - 数学, 334–336
 - 符号, 197, 334
 - 可变尺寸的符号, 335
 - 文本, 334
 - 数学字母表, 103, 196
 - Slanted sans serif 字体, 329
 - Slanted 字体, 327
 - slides 类, 208
 - 文件名, 336
 - 小大写字体, 327
 - 斜体, 327
 - 直立 sans serif 字体, 329
 - 直立斜体, 327
 - 装饰性字体, 324, 333
 - 字体放大, 337
- 字体尺寸, 58
 - 标准行间距, 58
 - 改变, 58, 191
 - 类选项, 33
 - 声明, 58
 - 在公式中, 118
- 字体的放缩, 61
- 字体定义文件, 198
- 字体环境, 59
- 字体样式
 - 强调, 57
 - 嵌套, 57
- 字体约定, 7
- 族 (字体属性), 59, 192